ЭЛЕКТРОННАЯ ВЫЧИСЛИТЕЛЬНАЯ МАШИНА "ЭЛЕКТРОНИКА 60"

ЭКСПЛУАТАЦИОННАЯ ДОКУМЕНТАЦИЯ

КНИГА 6

ЭЛЕКТРОННАЯ ВЫЧИСЛИТЕЛЬНАЯ МАШИНА
"ЭЛЕКТРОНИКА 60"
15 ВМ–16

ПЕРФОЛЕНТОЧНАЯ ОПЕРАЦИОННАЯ СИСТЕМА
ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ
0.005.027. ПО

© Центральный научно-исследовательский институт "Электроника" 1978

ВВЕДЕНИЕ

Настоящий документ содержит инструкции по использованию программ перфоленточной операционной системы для ЭВМ типа "Электроника 60", "Электроника 100-25".

Перфоленточная операционная система содержит следующие программы:

Перемещающий ассемблер
Ассемблер
Компоновщик
Редактор текста
Программа отладки
Программа начальной загрузки
Абсолютный загрузчик
Перфорация в абсолютном формате
Восьмеричная печать
Супервизор ввода-вывода

Используя инструкции на перечисленные выше программы, пользователь может загружать, выводить, редактировать, транслировать и отлаживать свои программы.

Для работы перфоленточной операционной системы необходимо следующее оборудование: процессор с емкостью оперативной памяти не менее 16К байт, электрифицированная пишущая машинка, высокоскоростные и/или низкоскоростные перфоленточные устройства ввода-вывода. Адреса регистров состояния используемых периферийных устройств приведены в табл.1.

Таблица 1

Адрес регистра состояния	Периферийное устройство
777514	высокоскоростное печатающее устройство (DZM-180)
777550	Высокоскоростное перфоленточное устройство ввода (FS 1501)
777554	Высокоскоростное перфоленточное устройство вывода (ИЛ-150)

Продолжение табл.I

Адрес регистра состояния	Периферийное устройство
777560	Клавиатура ЭПМ, низкоскоростное перфоленточное устройство ввода
777564	Печатающее устройство ЭПМ, низкоскоростное перфоленточное устройство вывода

В документе приняты сокращения, приведенные в табл.2.

Таблица 2

Сокращенное обозначение	Полное наименование
ДКС	Дополнение до контрольной суммы
ЗАН.АДР	Занесение адреса
ЗАП	Запись
КР	Клавишный регистр
ОПР	Операция
ПО	Программа отладки
ПРОД	Продолжение
РОН	Регистр(ы) общего назначения (0-7)
РСП	Регистр состояния программы
СК(R7)	Счетчик команд
СКЛ	Служебная клавиша
ССП	Слово состояния программы
ЧТЕН	Чтение
УС(R6)	Указатель стека
ЭПМ	Электрифицированная пишущая машинка

В документе приняты обозначения следующих символов:

- возврат каретки (ВК)
- перевод строки (ПС)

При использовании в качестве печатающего устройства ЭПМ "CONSUL 260" символам документа "¤" и "\" соответствуют символы "§" и "|" клавиатуры ЭПМ, символу "@" соответствует комбинация клавиши СКЛ и * (СКЛ/*)при вводе и "*" при выводе.

В документе приняты следующие правила:

а) символ или строка текста, вводимая пользователем, завершается нажатием клавиши ↵, если это не оговорено иначе;
б) знаки, выводимые ЭВМ на печатающее устройство ЭВМ, в отличие от вводимых пользователем данных, подчеркнуты.

Перед изучением данного документа необходимо ознакомиться с техническими описаниями на используемую ЭВМ и периферийные устройства.

I. ПЕРЕМЕЩАЮЩИЙ АССЕМБЛЕР

Данный раздел представляет собой руководство по использованию языка Перемещающего ассемблера.

Язык Перемещающего ассемблера является машинно-ориентированным языком. Трансляция исходной программы или модуля осуществляется на ЭВМ с емкостью оперативной памяти не менее 16К байт.

В процессе трансляции Перемещающий ассемблер осуществляет:

а) задание функций прохода и устройств ввода-вывода с помощью начального диалога;
б) печать сообщений об ошибках;
в) совмещение обработки программы с вводом-выводом;
г) печать таблицы имен пользователя в алфавитном порядке;
д) создание перемещаемых объектных модулей;
е) связь между объектными модулями с помощью глобальных имен;
ж) обработку участков программы с помощью директив условной трансляции;
з) секционирование программ.

Объектный модуль перфорируется при втором проходе. Он является входной информацией для программы "Компоновщик" (см.раздел 3).

Листинг - печатный документ исходного модуля - выдается при третьем проходе.

Трансляция выполняется за три прохода. Для высокоскоростных устройств ввода-вывода второй и третий проход совмещены.

I.I. Алфавит языка

Исходная программа на языке Перемещающего ассемблера составляется из имен, чисел, выражений, мнемоники команд и директив ассемблера, разделителей операндов и ограничителей строк, записываемых с использованием следующих символов:

а) прописные буквы латинского алфавита от A до Z;
б) арабские цифры от 0 до 9;
в) специальные знаки " . " и " ¤ ";
г) специальные знаки, используемые в качестве разделителей и ограничителей

: = % # @( ) , ; " ' + - & ! пробел .

д) символы формата

возврат каретки (ВК), перевод строки (ПС), перевод формата

(ПФ), горизонтальная табуляция (ГТ).

Примечание. Ассемблер допускает использование букв русского алфавита (А-Я) в комментариях, в операнде директивы .ASCII, а также при преобразовании одного иди двух знаков алфавита в соответствующие 7-битные коды.

1.2. Операторы

Исходная программа записывается в виде последовательности операторов. Каждый оператор занимает отдельную строку и ограничивается символом ВК, за которым должен следовать символ ПС или ПФ. Если за ВК не следует символ ПС или ПФ, то ассемблер выдает сообщение об ошибке "Q" (см.п.I.I6) и часть строки, следующая за символом ВК, игнорируется. Максимальная длина строки оператора -74 символа, включая ВК и ПС. Оператор может занимать от одного до четырех полей, которые распознаются по порядку их следования и ограничительным знакам. Это следующие поля:

метка: операция операнд; комментарий

Поля метки и комментария являются необязательными. Поля операции и операнда являются взаимозависимыми.

1.2.I. Метка

Метка представляет собой имя, определенное пользователем, которому присваивается текущее значение счетчика адресов. Требования, предъявляемые к меткам, см. в п.I.3. Значение метки может быть абсолютным или относительным в зависимости от вида значения счетчика адресов (см.п.I.5).

Метка должна стоять первой в операторе и ограничиваться знаком ":". Например, если текущее значение счетчика адресов есть абсолютное 000100

```
ABCD :  MOV A,B
```

то ассемблер присвоит это значение метке ABCD и последующее обращение к ABCD будет обращением к ячейке с адресом 000100. В поле метки может быть несколько меток; каждая метка внутри этого поля имеет одно и тоже значение. Например, если текущее значение счетчика адресов есть 000100 и в операторе содержится несколько меток:

```
ABC : DDD: A7.7 : MOV A,B
```

то каждой из трех меток ABC, DDD , A7.7 будет присвоено значение 000100 ("$" и " ." используются в метках для системных программ).

Если две или более меток имеют одни и те же первые 6 символов, то при трансляции печатается сообщение об ошибке "М".

I.2.2. Операция

В операторе за полем метки следует поле операции. Поле операции содержит мнемонику команды или директивы ассемблера (см.Приложение 2). Мнемоника команды определяет какие действия должны быть выполнены над операндом (операндами). Мнемоника директивы определяет функцию или действие, которое должно быть выполнено ассемблером в процессе трансляции.

Операции может предшествовать одна или несколько меток, а за ней следовать один или несколько операндов и/или комментарий. Поле операции ограничивается символом ГТ, пробелом или одним из следующих знаков:

\# + - @ (" ' % ! & , ;

Пример:

```
MOV  A,B ;      ГТ ОГРАНИЧИВАЕТ ОПЕРАЦИЮ   MOV
MOV@A,B ;       @ ОГРАНИЧИВАЕТ ОПЕРАЦИЮ    MOV
```

Когда операция записана без операнда или комментария, она ограничивается символом ВК, за которым следует символ ПС или ПФ.

I.2.3. Операнд

Операнд - это часть оператора, на которую действует операция - мнемоника команды или директивы ассемблера.

Операндом может быть имя, выражение или число. Если в операторе содержится несколько операндов, то их разделяют запятыми. Операнду предшествует операция и/или метка, а за ним может следовать комментарий. Поле операнда ограничивается точкой с запятой, если за ним следует комментарий, или символом ВК, за которым следует символ ПС или ПФ, если оператор заканчивается операндом. Например,

```
LABEL : MOV GEORGE, BOB; ЭТО КОММЕНТАРИЙ
```

Пробел между MOV и GEORGE ограничил поле операции и начал поле операнда; запятая разделила операнды GEORGE и BOB; точка с запятой ограничила поле операнда и начала поле комментария.

I.2.4. Комментарий

Поле комментария является необязательным и может содержать любой символ ЭПМ (см.приложение 1), за исключением символов ПУС, ЗБ, ВК, ПС и ПФ.

Полю комментария могут предшествовать одно, два или все три поля. Поле комментария должно начинаться с символа ";" и заканчиваться символом ВК, за которым следует символ ПС или ПФ. Например,

LABEL : CLR MERE ; ЭТО КОММЕНТАРИЙ

Комментарии не влияют на процесс трансляции или выполнение программы, но они полезны в листинге для последующего анализа, проверки и оформления документа.

1.2.5. Форматирование

Символ формата ГТ и пробел не влияют на процесс трансляции, если не включены внутрь имени, числа, операнда директивы .ASCII или взяты в качестве ограничителя поля операции. Они используются для наглядной записи программы. Например: оператор

LABEL:MOV(R6)+,TAG;ВЫБОРКА ИЗ СТЕКА

используя символ формата ГТ и пробел, может быть записан в форме

LABEL: MOV (R6)+, TAG ; ВЫБОРКА ИЗ СТЕКА

более удобной для чтения.

Ассемблер разбивает листинг на страницы по 56 строк в каждой и нумерует их. Нумерация страниц начинается с нуля. Если в исходной программе встречается символ ПФ, то формируется новая страница листинга.

I.3. Имена

Имя - это буквенно-цифровое изображение метки, команды или терма выражения.

К именам предъявляются следующие требования:

а) первый символ не должен быть цифрой;

б) каждое имя должно быть единственным по первым шести символам;

в) если имя записано более чем шестью символами, то седьмой и последующие символы будут проверяться ассемблером только на допустимость;

г) пробел и символ табуляции не должны включаться в имя.

Перемещающий ассемблер допускает два типа имен: постоянные имена и имена пользователя. Эти имена запоминаются в таблице имен ассемблера. Первоначально таблица имен содержит постоянные имена, в процессе трансляции имена пользователя добавляются в таблицу ассемблера.

I.3.I. Постоянные имена

Постоянные имена - мнемоника команд и директив ассемблера находятся в таблице имен ассемблера и не нуждаются в определении перед использованием в исходной программе.

I.3.2. Имена пользователя

Имена пользователя - это имена, которые определяются с помощью оператора прямого присваивания (см.п.I.3.3) или являются метками (см.п.I.2.I). Имена пользователя служат для обозначения адреса памяти или числовой константы. В процессе трансляции имена пользователя заносятся в таблицу имен ассемблера по мере их появления. Они могут содержать буквенно-цифровые символы, "$" и ". ". Любой другой символ является недопустимым и при использовании вызывает сообщение об ошибке "I".

Имя пользователя может дублировать постоянное имя. Значение, связанное с постоянным именем, зависит от его использования:

а) постоянному имени в поле операции соответствует машинный код операции;

б) постоянному имени в поле операнда соответствует определяемое пользователем значение. Если имя не определено пользователем, то его значением является соответствующий машинный код операции.

Имена пользователя могут быть глобальными или внутренними. Имена, описанные директивой .GLOBL (см.п.I.9.2), являются глобальными. Глобальные имена используются для организации связей между объектными модулями. Эти имена позволяют обращаться к другим объектным модулям. Глобальное имя, определенное в данном модуле (как метка или прямым присваиванием) называется именем входа или точкой входа. Глобальное имя, не определенное в данном объектном модуле, называется внешним именем. Оно должно быть определено в качестве точки входа в каком-то другом модуле. Имена, не описанные директивой .GLOBL, являются внутренними именами. Эти имена встречаются в текущем объектном модуле и обращение к ним происходит только внутри модуля.

I.3.3. Оператор прямого присваивания

Оператор прямого присваивания присваивает имени определенное значение. Когда оператор прямого присваивания определяет

имя в первый раз, то оно вводится в таблицу имен ассемблера вместе с его значением. Имя может быть переопределено путем задания нового значения. Новое значение заменяет значение имени присвоенное ранее.

Формат оператора прямого присваивания следующий:

имя = выражение

Имя принимает абсолютное или относительное значение в зависимости от вида выражения (см.п.I.4.4), которое ему присваивается. Однако, если выражение является глобальным, то имя глобальным не считается до тех пор, пока оно не будет описано директивой .GLOBL.

Применяются следующие правила:

а) знак "=" должен отделять имя от выражения, значение которого присваивается имени;

б) оператор прямого присваивания может следовать за меткой, а за ним, в свою очередь, может следовать комментарий;

в) только одно имя может быть определено одним оператором прямого присваивания;

г) допустим только один уровень ссылки вперед, например:

```
X = Y
Y = Z
Z = 1
```

X и Y в конце первого прохода будут напечатаны с признаком ****, который указывает, что имена не определены оператором прямого присваивания. При втором проходе неопределен X, это вызывает сообщение об ошибке "U"

Примеры:

```
A = 1                   ;   A ПРИСВАИВАЕТСЯ 1
B = 'A - 1&MASKLOW      ;   B ПРИСВАИВАЕТСЯ ЗНАЧЕНИЕ
                        ;   ВЫРАЖЕНИЯ 'A - 1& MASKLOW
C : D = 3               ;   D ПРИСВАИВАЕТСЯ 3, МЕТКАМ
E : MOV # 1, ABLE       ;   C И E ПРИСВАИВАЕТСЯ
                        ;   АДРЕС ЯЧЕЙКИ ПАМЯТИ, ПО КОТОРОМУ
                        ;   ЗАПИСАНА КОМАНДА MOV
```

I.3.4. Имена регистров

Обращение к регистру общего назначения осуществляется через его имя. Имя регистра определяется прямым присваиванием, где

определяемое выражение содержит по крайней мере один терм, которому предшествует знак % или один терм, предварительно определенный как имя регистра. Выражение для определения имени регистра должно быть абсолютным.

Примеры:

```
R0 = % 0        ;   ОПРЕДЕЛЯЕТСЯ R0 КАК РЕГИСТР 0
R3 = R0 + 3     ;   ОПРЕДЕЛЯЕТСЯ R3 КАК РЕГИСТР 3
R4 =  1 + % 3   ;   ОПРЕДЕЛЯЕТСЯ R4 КАК РЕГИСТР 4
THERE = % 2     ;   ОПРЕДЕЛЯЕТСЯ THERE  КАК
                ;   РЕГИСТР 2
```

Необходимо отметить, что все имена регистров должны быть определены до того как к ним производится обращение. Обращение к неопределенному имени регистра будет вызывать сообщение об ошибке "P". Обращение к регистру, номер которого не 0-7, будет вызывать сообщение об ошибке "R".

Знак % может быть использован в операторе, указывая на обращение к регистру.

Пример:

```
CLR     % 6
```

будет очищать регистр 6, в то время как оператор

```
CLR  6
```

будет очищать ячейку памяти с адресом 000006.

В некоторых случаях обращение к регистру может производиться без определения имени регистра или выражения регистра. Эти случаи распознаются по содержимому оператора.

Примеры:

```
JSR    5, SUBR  ;  ИСТОЧНИК ЯВЛЯЕТСЯ РЕГИСТРОМ
CLR  X(2)       ;  ЛЮБОЕ ВЫРАЖЕНИЕ, ЗАКЛЮЧЕННОЕ В
                ;  СКОБКИ, ВСЕГДА ЯВЛЯЕТСЯ РЕГИСТРОМ
                ;  В ДАННОМ СЛУЧАЕ В СКОБКАХ
                ;  РЕГИСТР 2
```

I.4. Выражения

Выражение – это терм или несколько термов, соединенных знаками арифметических и логических операций (см.п.I.4.2). Термом выражения может быть постоянное имя, имя пользователя, число или текущее значение счетчика адресов, обозначенное точкой. Выражения вычисляются слева направо и обрабатываются как слова. Группировка

в скобки является недопустимой. Выражение, полученное в результате обработки, может быть абсолютным, относительным или внешним (см.п.I.4.4).

Отсутствующий терм или выражение интерпретируется как нуль. Недостающий знак операции интерпретируется как "+" и вырабатывается сообщение об ошибке "Q".

Например:

```
A + - 100                ; ОТСУТСТВУЕТ ОПЕРАНД
будет вычисляться как A + 0 - 100
TAG ! LA  177777         ; ОТСУТСТВУЕТ ОПЕРАЦИЯ
будет вычисляться как TAG ! LA + 177777
```

I.4.I. Числа

Ассемблер допускает использование как восьмеричных, так и десятичных чисел. Восьмеричные числа состоят из цифр 0 - 7. Десятичные числа состоят из цифр 0 - 9 и должны заканчиваться точкой " . ", которую будем называть десятичной . Если число содержит цифру 8 или 9, но не заканчивается десятичной точкой, то вырабатывается сообщение об ошибке "N " и число интерпретируется как десятичное. Отрицательные числа представляются со знаком "-". Положительные числа могут быть представлены со знаком "+", хотя это и не обязательно. Числа всегда рассматриваются как абсолютные величины.

Если число занимает более 16 разрядов, оно усекается слева, т.е. разряды старше 16 игнорируются. При печати листинга число будет сопровождаться сообщением об ошибке "T".

I.4.2. Арифметические и логические операции

Арифметические и логические операции используются для записи выражений. Знаки арифметических операций следующие:

+ обозначает сложение или положительное число

- обозначает вычитание или отрицательное число.

Знаки логических операций следующие:

& обозначает логическую операцию "И"

! обозначает логическую операцию "ИЛИ".

I.4.3. Преобразование символов

Один или два символа ЭПМ, за исключением символов ПУС, ЗБ, ПС и ПФ, преобразуются ассемблером в 7-битные коды (см.Приложение 1) с помощью апострофа (') или кавычек ("). Для преобразова-

ния одного символа алфавита в соответствующий 7-битный код используемый апостроф.

Пример:

'A - символу A ассемблер присвоит значение 101_8

Для преобразования двух символов алфавита в соответствующие 7-битные коды используются кавычки. Для записи 7-битного кода требуется один байт, байты объединяются в слова. Например "AB вызовет запоминание кода A в младшем (четном) байте слова и кода B - в старшем (нечетном) байте слова.

```
Нечетный байт                 |     Четный байт
Значение B = 1 0 2            |     1 0 1 = Значение A
          0 100001 0          |   0 1000 001
          0  4   1            |   1   0   1
          " AB = 041101
```

Значение преобразованных символов всегда абсолютно.

I.4.4. Виды выражений

Выражение может быть абсолютным, относительным или внешним. Терм выражения является абсолютным, относительным или внешним в соответствии с тем, как он определен. Числа, постоянные имена и сформированные данные всегда интерпретируются как абсолютные.

Абсолютное выражение определяется как

а) абсолютный терм, перед которым необязательно стоит знак плюс или минус;

б) относительное выражение минус относительный терм, где оба относятся к одной и той же программной секции;

в) два абсолютных выражения, связанные знаком операции.

Относительное выражение определяется как

а) относительный терм;

б) относительное и абсолютное выражение, связанные знаком арифметической операции;

в) абсолютное выражение, связанное знаком операции сложения с относительным выражением.

Внешнее выражение определяется как

а) внешний терм;

б) внешнее выражение, связанное знаком арифметической операции с абсолютным термом;

в) абсолютное выражение, связанное знаком операции сложения с внешним выражением.

В следующих примерах

ABS - абсолютное имя

REL - относительное имя

EXT - внешнее имя

Примеры:

Верные выражения:

```
EXT + ABS                ; ВНЕШНЕЕ ВЫРАЖЕНИЕ
REL + REL - REL          ; ОТНОСИТЕЛЬНОЕ ВЫРАЖЕНИЕ
ABS + REL - REL & ABS    ; АБСОЛЮТНОЕ ВЫРАЖЕНИЕ
```

Неверные выражения:

```
EXT + REL
REL + REL
ABS - EXT
```

I.5. Счетчик адресов

Точка " . " представляет собой символ для обозначения счетчика адресов. Когда символ "." используется в поле операнда операции, он представляет адрес первого слова команды. Когда он используется в поле операнда директивы ассемблера, он представляет адрес текущего байта или слова.

Пример:

```
A : MOV #. , R0     ; ТОЧКА УКАЗЫВАЕТ НА АДРЕС А
```

(использование знака " # " дано в п.I.7.9).

В начале каждого прохода перемещающий ассемблер очищает счетчик адресов. В процессе трансляции счетчик адресов принимает значения последовательных адресов памяти. Однако, значение счетчика адресов может быть изменено с помощью оператора прямого присваивания:

. = выражение

Выражение, определяющее счетчик адресов, не должно содержать ссылок вперед (т.е. имен, которые определяются в программе позже) или имен, значения которых изменяются от прохода к проходу.

Подобно именам, символ счетчика адресов и соответственно его значение может быть абсолютным или относительным. Этот символ не может быть внешним. Изменить вид значения символа счетчика адресов можно с помощью директивы .ASECT или .CSECT (см.п.I.9).

Примеры:

```
  .ASECT                 ; УСТАНАВЛИВАЕТ В СЧЕТЧИКЕ
```

```
      . = 500                  ; АДРЕСОВ АБСОЛЮТНОЕ ЗНАЧЕНИЕ 500
FIRST: MOV .+ 10, COUNT        ; МЕТКА FIRST ИМЕЕТ АБСОЛЮТНОЕ
                               ; ЗНАЧЕНИЕ 500
                               ; .+10 ПРИСВАИВАЕТСЯ К 510
                               ; СОДЕРЖИМОЕ ЯЧЕЙКИ 510 БУДЕТ
                               ; ЗАПИСАНО В ЯЧЕЙКУ COUNT
      .=520                    ; СЧЕТЧИК АДРЕСОВ ИМЕЕТ АБСОЛЮТНОЕ
                               ; ЗНАЧЕНИЕ 520
SECOND: MOV .,INDEX            ; МЕТКА SECOND  ИМЕЕТ АБСОЛЮТНОЕ
                               ; ЗНАЧЕНИЕ 520
                               ; СОДЕРЖИМОЕ ЯЧЕЙКИ 520 (ДВОИЧНЫЙ КОД
                               ; САМОЙ КОМАНДЫ) БУДЕТ ЗАПИСАНО В
                               ; ЯЧЕЙКУ INDEX
      .CSECT                   ; УСТАНАВЛИВАЕТ В СЧЕТЧИКЕ АДРЕСОВ
      .=.+ 20                  ; ОТНОСИТЕЛЬНОЕ ЗНАЧЕНИЕ 20
THIR : .WORD 0                 ; МЕТКА THIR ИМЕЕТ ОТНОСИТЕЛЬНОЕ
                               ; ЗНАЧЕНИЕ 20
```

Можно зарезервировать область памяти путем увеличения значения счетчика адресов. Например, если текущее значение счетчика равно 001000, то оператор прямого присваивания

. = . +100

зарезервирует 100_8 байтов памяти. Следующая команда будет запоминаться в ячейке с адресом 001100.

I.6. Перемещение и связка

Перемещающий ассемблер создает объектный модуль, из которого программа "Компоновщик" получает загрузочный модуль, присваивая абсолютные значения внешним и относительным именам. Загрузочный модуль - это программа в двоичном коде, которую можно загрузить в память ЭВМ и выполнить. Для того чтобы Компоновщик присваивал абсолютные значения выражениям, Перемещающий ассемблер в объектном модуле выдает ряд директив с требуемыми параметрами. В случае относительного выражения Компоновщик прибавляет значение относительного выражения, полученного при трансляции программы, к величине смещения программы. В случае внешнего выражения Компоновщик определяет значение внешнего терма выражения и эта величина прибавляется к значению внешнего выражения, получаемого при трансляции программы. При этом внешнее имя должно определять-

ся в одном из других объектных модулей, связанных между собой. Все выражения, которые модифицируются как указано выше, при получении листинга снабжаются апострофом (').

Примеры:

```
005065' CLR EXTERNAL (5)    ; ЗНАЧЕНИЕ ВНЕШНЕГО
000000                      ; ИМЕНИ EXTERNAL, ВЫДАННОГО
                            ; АССЕМБЛЕРОМ КАК НУЛЬ, БУДЕТ
                            ; МОДИФИЦИРОВАТЬСЯ КОМПОНОВЩИКОМ
005065' CLR EXTERNAL +6(5) ; АБСОЛЮТНАЯ ЧАСТЬ
000006                      ; ВЫРАЖЕНИЯ (000006)
                            ; ПРИБАВЛЯЕТСЯ КОМПОНОВЩИКОМ К
                            ; ЗНАЧЕНИЮ ВНЕШНЕГО ИМЕНИ
005065' CLR RELOCATABLE(5) ; ЗНАЧЕНИЕ ОТНОСИТЕЛЬНОЙ ВЕЛИЧИНЫ
000040                      ; ЕСТЬ 40,
                            ; МЫ НАХОДИМСЯ В АБСОЛЮТНОЙ СЕКЦИИ
                            ; ПРОГРАММЫ
```

I.7. Методы адресации

Счетчик команд (СК - регистр общего назначения R7) всегда содержит адрес слова, которое должно быть выбрано, т.е. адрес второго или третьего слова текущей команды или адрес следующей команды. Содержимое счетчика команд после выборки слова из памяти автоматически увеличивается на 2.

Введем следующие обозначения:

а) E - выражение, определенное в п.I.4 ;

б) R - выражение регистра (см.п.I.3.4); .

в) ER - выражение регистра, определенное в пределах 0÷7;

г) A - операнд, метод адресации которого может быть выражен с помощью обозначений E, R и ER.

I.7.I. Регистровый метод

Формат : R

Регистр содержит операнд.

Примеры:

```
R0   = % 0      ; ОПРЕДЕЛИТЬ R0 КАК РЕГИСТР 0
CLR  R0         ; ОЧИСТИТЬ R0
```

I.7.2. Косвенно-регистровый метод

Формат: @R или (ER)

Регистр содержит адрес операнда.

Примеры:

```
CLR @R1          ; ОЧИСТИТЬ ЯЧЕЙКУ, АДРЕС
  или
CLR  (1)         ; КОТОРОЙ НАХОДИТСЯ В R1
```

I.7.3. Автоинкрементный метод

Формат: (ER)+

Регистр содержит адрес операнда.

Содержимое регистра, после его использования как адреса операнда, автоматически увеличивается.

Примеры:

```
CLR (R0)+        ; ОЧИСТИТЬ СОДЕРЖИМОЕ ЯЧЕЕК,
CLR (R0+3)+      ; АДРЕСА КОТОРЫХ НАХОДЯТСЯ
CLR (2)+         ; В РЕГИСТРАХ 0,3,2 И УВЕЛИЧИТЬ
                 ; СОДЕРЖИМОЕ РЕГИСТРОВ НА 2
```

I.7.4. Косвенно-автоинкрементный метод

Формат: @(ER)+

Регистр содержит указатель адреса операнда. Содержимое регистра увеличивается после выборки указателя адреса операнда.

Пример:

```
CLR @(3)+        ; СОДЕРЖИМОЕ РЕГИСТРА 3
                 ; УВЕЛИЧИВАЕТСЯ НА 2
                 ; ПОСЛЕ ВЫБОРКИ УКАЗАТЕЛЯ
                 ; АДРЕСА ОПЕРАНДА
```

I.7.5. Автодекрементный метод

Формат: -(ER)

Регистр содержит адрес операнда. Содержимое регистра автоматически уменьшается, а затем используется как адрес операнда.

Примеры:

```
CLR -(R0)        ; СОДЕРЖИМОЕ РЕГИСТРОВ 0, 3 И 2
CLR -(R0 +3)     ; УМЕНЬШАЕТСЯ ПЕРЕД ВЫБОРКОЙ
CLR -(2)         ; АДРЕСА ОПЕРАНДА
```

I.7.6. Косвенно-автодекрементовый метод

Формат: @-(ER)

Регистр содержит указатель адреса операнда. Содержимое регистра автоматически уменьшается, а затем используется как указатель адреса операнда.

Пример:

```
CLR @-(2)          ; СОДЕРЖИМОЕ РЕГИСТРА 2
                   ; УМЕНЬШАЕТСЯ ДО ИСПОЛЬЗОВАНИЯ
                   ; ЕГО КАК УКАЗАТЕЛЯ АДРЕСА ОПЕРАНДА
```

I.7.7. Индексный метод

Формат: E(ER)

Значение E (индексное слово) запоминается как второе или третье слово команды. Адрес операнда вычисляется как сумма значений E и ER.

Примеры:

```
CLR X+2(R1)        ; ИСПОЛНИТЕЛЬНЫЙ АДРЕС
                   ; РАВЕН X + 2 ПЛЮС СОДЕРЖИМОЕ R1
CLR -2(3)          ; ИСПОЛНИТЕЛЬНЫЙ АДРЕС РАВЕН -2 ПЛЮС
                   ; СОДЕРЖИМОЕ R3
```

I.7.8. Косвенно-индексный метод

Формат: @E(ER)

Значение E и содержимое регистра, определяемого выражением ER, складываются, и сумма используется как указатель адреса операнда.

Пример:

```
CLR @14(4)         ; ЕСЛИ В РЕГИСТРЕ 4 НАХОДИТСЯ 100,
                   ; В ЯЧЕЙКЕ 114 НАХОДИТСЯ 2000, ТО
                   ; ЯЧЕЙКА 2000 ОЧИЩАЕТСЯ
```

I.7.9. Непосредственный метод адресации

Формат: #E

Операнд запоминается во втором или третьем слове команды. Непосредственный метод формируется как автоинкрементный метод адресации с использованием СК.

Пример:

```
MOV #100, R3       ; ЗАНЕСТИ 100 В R3
```

Оператор MOV # 100, R3 транслируется в два слова:

012703

000100

Перед выборкой и выполнением этой команды СК указывает на адрес первого слова команды. Процессор выбирает первое слово и увеличивает содержимое СК на 2. Метод адресации операнда источника - 27. Таким образом, СК используется как указатель для выборки операнда (второго слова команды) перед увеличением его содержимого на 2 для указания адреса следующей команды.

I.7.IO. Абсолютный метод адресации

Формат: @ # E

Абсолютный адрес операнда запоминается во втором или третьем слове команды.

Пример:

```
CLR  @ # 100        ; ОЧИСТИТЬ ЯЧЕЙКУ 100
```

Этот оператор будет очищать ячейку с адресом 000100.

I.7.II. Относительный метод адресации

Формат: E

Адрес операнда задается с помощью СК и индексного слова.

Примеры:

```
CLR  100            ; ОЧИСТИТЬ ЯЧЕЙКУ 100
MOV X, R0           ; ЗАНЕСТИ СОДЕРЖИМОЕ ЯЧЕЙКИ С АДРЕСОМ X
                    ; В РЕГИСТР R0
```

Этот метод организуется как индексный метод, используя в качестве регистра СК. Индексное слово вычисляется как E-СК и хранится во втором или третьем слове команды. Индексное слово является числом, которое прибавляется к содержимому СК для образования исполнительного адреса операнда.

Пример:

Оператор MOV 100, R3 транслируется по адресу 000020

```
000020  : 016703
000022  : 000054
```

Процессор выбирает команду MOV и увеличивает СК на 2, указывая на ячейку 000022. Метод адресации операнда источника - 67, т.е. индексный метод адресации, где в качестве регистра используется СК. Значение индексного слова процессор выбирает по адресу, указанному СК, а затем увеличивает СК на 2. Таким образом

в СК находится адрес 000024. Для вычисления адреса операнда источника значение индексного слова прибавляется к содержимому СК, т.е. адрес операнда равен:
ИНДЕКСНОЕ СЛОВО + СК = 54+24 = 100

Так как ассемблер рассматривает точку как адрес первого слова команды, то оператор запишется так:
MOV 100 -.-4 (R7), R3.

Этот метод адресации называется относительным, так как адрес операнда вычисляется относительно текущего значения СК. Если оператор и его операнд перемещаются в памяти так, что расстояние между оператором и операндом остается постоянным, то команда будет выполняться правильно.

I.7.I2. Косвенно-относительный метод адресации

Формат: @E

Указатель адреса операнда задается с помощью СК и индексного слова. Эта адресация аналогична относительной адресации, только сформированный адрес является указателем адреса операнда.

I.8. Форматы команд

Приведем структуру используемых команд (см.Приложение 2) в табл.3. Предварительно дадим соответствие прямых и косвенных методов адресации.

E	@E
R	@R или (R)
(ER)+	@(ER)+
-(ER)	@- (ER)
E(ER)	@ E (ER)
# E	@ # E

Таблица 3

Команда	Формат команды	Пример
Двухадресная	ОПР A, A	MOV (R6) +,@V
Одноадресная	ОПР A	CLR -(R2)
Безадресная	ОПР	HALT
Ветвление	ОПР E , где $-128_{10} \leq (E-.-2)/2 \leq 127_{10}$	BR X+2 BLO .-4
Обращение к подпрограмме	JSR ER, A	JSR R7, SUBR

Продолжение табл.3

Команда	Формат команды	Пример
Возврат из подпрограммы EMT/TRAP	RTS ER ОПР или ОПР E , где $0 \le E \le 377_8$	RTS R7 EMT или EMT 31

Команды ветвления состоят из одного командного слова. Старший байт слова содержит код операции, младший - восьмиразрядное смещение в дополнительном коде, которое определяет адрес перехода относительно содержимого СК. Исполнительный адрес аппаратно вычисляется следующим образом:

а) расширяется знак смещения с разряда 8 по 15-ый;
б) результат умножается на 2;
в) результат прибавляется к содержимому СК для формирования исполнительного адреса.

Ассемблер выполняет обратную операцию для формирования смещения в байтах относительно заданного адреса. Смещение в байтах равно (E-СК) /2 или (E-.-2) /2, усеченное до 8 разрядов (СК =.+2).

Командами EMT и TRAP не используется младший байт команды. Это позволяет задавать в младшем байте команды номер подпрограммы, обслуживающей внутренние прерывания. Если в командах EMT и TRAP операндом является выражение, то его результат должен быть меньше 377_8. Если результат выражения больше 377, то он усекается до 8 разрядов и на печатающее устройство ЭПМ выдается сообщение об ошибке "T".

I.9. Директивы ассемблера

Директивы ассемблера управляют процессом трансляции исходных программ и формированием данных. Мнемоника директивы начинается с точки и записывается в поле операции. Поле операнда может быть пустым или содержит один или несколько операндов. Перед директивой может стоять метка, а за директивой следовать комментарий. По своему назначению директивы ассемблера делятся на:

- директиву присвоения наименования объектному модулю
  .TITLE

- директиву описания имен
  .GLOBL
- директивы секций
  .ASECT
  .CSECT
- директивы записи данных
  .BYTE
  .WORD
- директивы преобразования символов
  .ASCII
  .RAD50
- директиву границ программы
  .LIMIT
- директиву управления счетчиком адресов
  .EVEN
- директивы окончания
  .EOT
  .END
  .ENDC
- директивы трансляции по условию
  .IFZ E
  .IFNZ E
  .IFL E
  .IFLE E
  .IFG E
  .IFGE E
  .IFDF S(1) [!,&] S(2)[!,&]...[!,&]S(N)
  .IFNDF S(1) [!,&] S(2)[!,&]...[!,&]S(N)

где !, & - знаки логических операций.

I.9.I. Директива .TITLE

Директива .TITLE используется для присвоения наименования объектному модулю.

Формат директивы:

.TITLE NNNNNN ,

где NNNNNN - наименование объектного модуля.

Наименование может состоять из шести допустимых символов. Если директива .TITLE отсутствует, то объектному модулю ассемблер присваивает наименование .MAIN . Если в исходном модуле встречает-

ся более чем одна директива .TITLE, то объектному модулю присваивается наименование, заданное последней директивой .TITLE.

I.9.2. Директива .GLOBL

Директива .GLOBL используется для описания имен как глобальных.

Формат директивы:

.GLOBL NAMA, NAMB, ..., NAMN ,

где NAMA, NAMB, ..., NAMN- глобальные имена.

Описываемые имена разделяются запятыми или пробелами. Глобальные имена могут быть именами входа или внешними именами.

Пример:

```
        .CSECT
        .GLOBL   A,B,C          ; ОПРЕДЕЛЯЕТ A, B И C КАК
                                ; ГЛОБАЛЬНЫЕ ИМЕНА
A:      MOV@(R5)+,R0            ; ТОЧКА ВХОДА A
        MOV # X, R1
X:      JSR   R7, C             ; ИСПОЛЬЗОВАНИЕ ВНЕШНЕГО ИМЕНИ
        RTS  R5
B:      MOV@(R5)+, R1           ; ТОЧКА ВХОДА  B
        CLR   R1
        BR    X
```

В данном примере A и B - имена входа, C - внешнее имя, X - внутреннее имя.

I.9.3. Директивы .ASECT и .CSECT

Директивы секций дают возможность пользователю делить программу и соответственно объектные модули на секции: абсолютную, определяемую директивой .ASECT, и относительную, определяемую директивой .CSECT.

Формат директив

.ASECT

.CSECT

Если директива секции не задана, то ассемблер предполагает заданной директиву .CSECT. Для каждой секции ассемблер использует отдельные счетчики адресов. Первоначально счетчики адресов содержат абсолютный и относительный нуль. Сфера действия каждой директивы простирается до появления другой директивы секции. Дальнейшее появление директивы .CSECT или .ASECT продолжает

трансляцию программы с места окончания соответствующей секции.

Пример:

```
        .CSECT          ; НАЧАЛО ОТНОСИТЕЛЬНОЙ СЕКЦИИ
A:      0               ; ТРАНСЛИРУЕТСЯ ПО ОТНОСИТЕЛЬНОМУ АДРЕСУ 0
B:      0               ; ПО ОТНОСИТЕЛЬНОМУ АДРЕСУ  2
C:      0               ; ПО ОТНОСИТЕЛЬНОМУ АДРЕСУ  4
ST:     CLR  A          ; ТРАНСЛИРУЕТСЯ ПО ОТНОСИТЕЛЬНЫМ АДРЕСАМ
        CLR  B          ; С 6 ПО 21
        CLR  C
        .ASECT          ; НАЧАЛО АБСОЛЮТНОЙ СЕКЦИИ
        . = 4
        .WORD .+2, HALT ; ТРАНСЛИРУЕТСЯ ПО  АБСОЛЮТНЫМ
                        ; АДРЕСАМ С 4 ПО 7
        .CSECT          ; ПРОДОЛЖАЕТ ОТНОСИТЕЛЬНУЮ СЕКЦИЮ
        INC  A          ; ТРАНСЛИРУЕТСЯ ПО ОТНОСИТЕЛЬНЫМ
                        ; АДРЕСАМ С 22 ПО 27
        .END            ; КОНЕЦ
```

Символ счетчика адресов "." является относительным или абсолютным в зависимости от того к какой секции он относится. Значение счетчика адресов можно изменить, если за директивой секции будет следовать оператор прямого присваивания, с помощью которого символу счетчика адресов присваивается новое числовое значение. Все метки в абсолютной секции являются абсолютными. Все метки в относительной секции являются относительными. Любым меткам, появляющимся в строке с директивой .ASECT или .CSECT, присваивается текущее значение счетчика адресов предыдущей секции. Например, если первая строка программы есть:

```
A :   .ASECT
```

то метке A присваивается значение относительного нуля, так как ассемблер начинает трансляцию программы с относительной секции, если директива секции не указана. Неопределенным внутренним именам присваивается значение нуля относительного или абсолютного.

Во время трансляции программы неизвестно куда должна быть загружена относительная секция программы. Взаимные обращения между секциями одной программы транслируются ассемблером относительно начального адреса этой секции. Ассемблер обеспечивает программу "компоновщик" необходимой информацией для правильной

связки. В этом нет необходимости при обращении к абсолютной секции (ассемблер знает все адреса загрузки для абсолютной секции).

Пример:

```
     .ASECT
     . = 1000
A :  CLR  X          ; ТРАНСЛИРУЕТСЯ КАК CLR ОТ НАЧАЛЬНОГО
                     ; АДРЕСА ОТНОСИТЕЛЬНОЙ СЕКЦИИ + 10
     JMP  Y          ; ТРАНСЛИРУЕТСЯ КАК  JMP ОТ НАЧАЛЬНОГО
                     ; АДРЕСА ОТНОСИТЕЛЬНОЙ СЕКЦИИ +6
     .CSECT
     MOV  R0, R1
     JMP  A          ; ТРАНСЛИРУЕТСЯ КАК JMP 1000
Y :  HALT
X :  0
     .END
```

Пример:

```
     .CSECT          ; ТРАНСЛИРУЕТСЯ ПО ОТНОСИТЕЛЬНЫМ
     .WORD 0,1,2     ; АДРЕСАМ 0,2 и 4
     .ASECT
     .WORD 0,1,2     ; ТРАНСЛИРУЕТСЯ ПО АБСОЛЮТНЫМ
                     ; АДРЕСАМ 0,2 и 4
     .CSECT
     .WORD 0         ; ТРАНСЛИРУЕТСЯ ПО ОТНОСИТЕЛЬНОМУ
                     ; АДРЕСУ 6
     .END
```

I.9.4. Директива .BYTE

Директива .BYTE используется для записи данных операндов в виде последовательно расположенных байтов.

Директива .BYTE может иметь один или несколько операндов, отделенных друг от друга запятыми. Операндом может быть любое допустимое выражение, результат которого занимает не более 8 разрядов. Если результат выражения занимает более 8 разрядов, то он усекается до 8 младших разрядов и выдается сообщение об ошибке "T"

Пример:

```
SAM = 5
. = 410
```

```
.BYTE  48., SAM      ; В ЯЧЕЙКЕ 410 ЗАПОМИНАЕТСЯ
                     ; 060 (ВОСЬМЕРИЧНЫЙ КОД 48),
                     ; В ЯЧЕЙКЕ 411 ЗАПОМИНАЕТСЯ 005
```

Директива .BYTE, за которой следует один или несколько пустых операндов, отделенных друг от друга запятыми, записывает нули для этих операндов.

Пример:

```
. = 420
.BYTE  , ,           ; НУЛЬ БУДЕТ ЗАПОМИНАТЬСЯ В
                     ; ЯЧЕЙКАХ 420, 421, 422
```

Если операндом является относительное выражение, то выдается сообщение об ошибке "A", предупреждающее, что во время связки перемещение, возможно, приведет к выражению, содержащему более 8 разрядов.

I.9.5. Директива .WORD

Директива .WORD используется для записи данных (операндов) в виде последовательно расположенных слов. Директива .WORD может иметь один или несколько операндов, отделенных друг от друга запятыми. Операндами могут быть любые допустимые выражения, результаты которых занимают не более 16 разрядов. Если результат выражения занимает более 16 разрядов, то он усекается до 16 разрядов слева и выдается сообщение об ошибке "T".

Пример:

```
. = 1420
SAL= 0
.WORD 177535 , .+4, SAL  ; В ЯЧЕЙКАХ 1420, 1422,
                         ; 1424 ЗАПОМИНАЮТСЯ
                         ; 177535, 1426 И 0
```

Директива .WORD, за которой следует один или несколько пустых операндов, отделенных друг от друга запятыми, записывает нули для этих операндов.

Примеры:

```
    . = 1430             ; В ЯЧЕЙКАХ 1430, 1432,
    .WORD , 5,           ; 1434 ЗАПОМИНАЮТСЯ 0, 5И0
    . = 440
LABEL : + MOV, LABEL     ; КОД КОМАНДЫ MOV (010000)
                         ; ЗАПОМИНАЕТСЯ В ЯЧЕЙКЕ 440
                         ; 440 ЗАПОМИНАЕТСЯ В ЯЧЕЙКЕ 442
```

Всякий раз, когда встречается имя, не являющееся постоянным именем, или, когда встречается постоянное имя, перед которым стоит знак арифметической или логической операции, ассемблер интерпретирует оператор как директиву .WORD. Поэтому, если мнемоника команды или директивы ассемблера записана с ошибками, то ассемблер воспринимает ее как директиву .WORDи выдает сообщение об ошибке. Например, имя MOV неправильно записано:

```
MOR  A, B
```

В результате будут выдаваться сообщения о двух ошибках: "Q", поскольку знак операции между MOR и A пропущен, и "U", так как имя MOR не определено.

I.9.6. Директива .ASCII

Директива .ASCII используется для преобразования последовательности символов ЭВМ в их 7-битные коды, за исключением символа ЗБ и символов, коды которых 00-37.

Формат директивы

.ASCII / $C_1C_2C_3$ ... C_N / ,

где $C_1C_2C_3$ . . .C_N - последовательность символов;

/ / - ограничители.

Ограничителями могут быть любые символы, за исключением двоеточия, знака равенства и символов, записанных в операнде директивы. 7-битные коды символов запоминаются в виде последовательно расположенных байтов объектной программы.

Пример:

```
. = 500
.ASCII /YES/        ; КОДЫ БУКВ Y,E,S ЗАПОМИНАЮТСЯ
                    ; СООТВЕТСТВЕННО В ЯЧЕЙКАХ 500, 501, 502
.ASCII /5+3/2/      ; КОДЫ СИМВОЛОВ 5, +, 3 ЗАПОМИНАЮТСЯ
                    ; СООТВЕТСТВЕННО В ЯЧЕЙКАХ 503, 504, 505
                    ; СИМВОЛ 2 АССЕМБЛЕРОМ НЕ
                    ; ВОСПРИНИМАЕТСЯ ТАК КАК СРЕДИ
                    ; ПОСЛЕДОВАТЕЛЬНОСТИ СИМВОЛОВ
                    ; НАХОДИТСЯ ОГРАНИЧИТЕЛЬНЫЙ ЗНАК "/"
```

Мнемоника директивы .ASCII должна быть ограничена пробелом или символом ГТ.

I.9.7. Директива .RAD50

Директива .RAD50 предназначена для преобразования символов в форму RAD50.

Формат директивы:

.RAD50 / $C_1C_2C_3$/ ,

где $C_1C_2C_3$ - операнд директивы;
/ / - ограничители.

Операндом директивы могут быть буквы латинского алфавита от A до Z, арабские цифры от 0 до 9, пробел и специальные знаки "." и "$". Ограничителями могут быть любые печатные знаки, за исключением знаков "=" и ":". Директива .RAD50 производит упаковку трех символов в одно слово:

код RAD50 = ($(C_1 \cdot 50_8) + C_2$) $50_8 + C_3$

Если операнд содержит менее трех символов, то недостающие символы дополняются пробелами.

Пример:

```
.RAD50 /ABC/          ; УПАКОВКА ABC В ОДНО СЛОВО
.RAD50 /AB/           ; УПАКОВКА AB И ПРОБЕЛА В ОДНО СЛОВО
.RAD50 /  /           ; УПАКОВКА В ОДНО СЛОВО ТРЕХ
                      ; ПРОБЕЛОВ
```

Коды символов в RAD50 приведены в Приложении 2.

I.9.8. Директива .LIMIT

В некоторых случаях надо знать границы перемещаемых (относительных) кодов модуля загрузки. Директива .LIMIT резервирует два слова, в которые программа "Компоновщик" засылает младший и старший адреса относительного кода модуля загрузки. Младший адрес (вводится в первое слово) является адресом первого байта относительного кода. Старший адрес является адресом первого свободного байта, следующего за относительным кодом. Эти адреса всегда четные. Относительная секция загружается по четным адресам. Если относительная секция состоит из нечетного числа байтов, то программа "Компоновщик" увеличивает это число на единицу, чтобы старший адрес был четным.

I.9.9. директива .EVEN

директива .EVEN обеспечивает четность содержимого счетчика адресов добавлением единицы в случае его нечетности. Операнды,

следующие за директивой .EVEN, игнорируются.

I.9.IO. Директива .EOT

Директива .EOT указывает на физический конец ленты. Если исходный модуль или программа находится на нескольких лентах, то каждая лента должна заканчиваться директивой .EOT. За директивой .EOT должны следовать по крайней мере два символа ПС или ПФ. Если за .EOT следует один символ ПС или ПФ, то ассемблер загружает всю ленту, но не обрабатывает символы первой строки, следующей за .EOT. Последняя лента должна заканчиваться директивой .END.

I.9.II. Директива .END

Директива .END указывает на логический и физический конец исходного модуля. За директивой .END может следовать операнд - пусковой адрес программы. Программа пользователя после загрузки будет выполняться с указанного адреса.

I.9.I2. Директивы условной трансляции

Директивы условной трансляции позволяют включать или исключать из процесса трансляции участки исходного текста.

Формат директив:

```
.IFZ  E          ; ЕСЛИ E = 0
.IFNZ E          ; ЕСЛИ E ≠ 0
.IFL  E          ; ЕСЛИ E < 0
.IFLE E          ; ЕСЛИ E ≤ 0
.IFG  E          ; ЕСЛИ E > 0
.IFGE E          ; ЕСЛИ E ≥ 0
.IFDF  S(1) [!,&]S(2)[!,&]...[!,&] S(N)
.IFNDF S(1) [!,&]S(2)[!,&]...[!,&] S(N)    ,
```

где ! и & - знаки логических операций,

E - выражение,

S(N) - символ.

Директива .IFDF означает "если определено", директива .IFNDF - "если неопределено".

Если условие выполняется, то транслируются все операторы, следующие за директивой условной трансляции до директивы .ENDC, где .ENDC - есть директива окончания условной трансляции. В противном случае операторы игнорируются до появления .ENDC. Обработка выражения, определяющего условие директивы, идет слева направо, заключение в скобки не разрешается.

Пример:

```
.IFDF   S!T&U           ; ТРАНСЛИРУЕТСЯ ПРИ УСЛОВИИ ,
                        ; ЧТО S ИЛИ T  ОПРЕДЕЛЕНЫ И
                        ; ОПРЕДЕЛЕНО U
.IFNDF  T&U!S           ; ТРАНСЛИРУЕТСЯ ПРИ УСЛОВИИ,
                        ; ЧТО T И U НЕ ОПРЕДЕЛЕНЫ
                        ; ИЛИ  S  НЕ ОПРЕДЕЛЕНО
```

Ошибочное или нулевое выражение вызывает замену недостающей величины нулем для проверки выполнения по условию.

В выражениях допускаются ограничители ";", "!", "&" или ВК. Другие ограничители, а также ошибочные и нулевые символы приводят к неопределенной ситуации для .IFDF и .IFNDF. Выражения, находящиеся в поле операнда директивы .ENDC, игнорируются. Метки в директивах условной трансляции разрешаются.

Пример:

```
A :   .IFZ 1
          :
B :   .ENDC
```

Метка A вводится в таблицу имен. Метка B - игнорируется, поскольку условие трансляции не выполняется. Если директива .END встречается внутри условия, которое выполняется, то появляется сообщение об ошибке "Q" и выполняется директива .END. Допускается вложение директив условной трансляции на глубину до 127_{10}. Если директив .ENDC в программе больше, чем директив условной трансляции, для лишних директив выдается сообщение об ошибке "Q".

I.IO. Трансляция исходной программы

Программа пользователя может состоять из нескольких исходных модулей. Трансляция каждого модуля осуществляется отдельно. При первом проходе создается таблица имен пользователя и выдается на ЭПМ список неопределенных имен. При втором проходе формируется и перфорируется объектный модуль. При третьем проходе печатается листинг. Таблица имен пользователя и сообщения об ошибках могут быть выданы при любом проходе. При работе с высокоскоростными устройствами ввода-вывода второй и третий проход совмещены. Устройства ввода-вывода и режим работы ассемблера задаются во время начального диалога.

I.11. Загрузка ассемблера

Перфолента "Перемещающий ассемблер" загружается в память ЭВМ с помощью Абсолютного загрузчика в нормальном режиме (см.п.6.2).

Абсолютный загрузчик сначала загружает часть ассемблера, которая очищает память до начального адреса Абсолютного загрузчика, затем загружает остальную часть ассемблера.

I.12. Начальный диалог

После загрузки Перемещающего ассемблера или его повторного запуска передается управление на начальный диалог. Вопросы ассемблера и ответы пользователя приведены в табл.4,5.

Таблица 4

Символ, печатаемый ассемблером	Значение символа
ж S	Какое устройство используется для ввода исходного модуля?
ж B	Какое устройство используется для вывода объектного модуля?
ж L	Какое устройство используется для выдачи листинга?
ж T	На какое устройство должна выдаваться таблица имен пользователя?

Таблица 5

Символ, печатаемый пользователем	Значение символа
T	ЭПМ
L	Низкоскоростное перфоленточное устройство ввода-вывода
H	Высокоскоростное перфоленточное устройство ввода-вывода
P	Высокоскоростное печатающее устройство
/1	Первый проход
/2	Второй проход

Продолжение табл.5

Символ, печатаемый пользователем	Значение символа
/3	Третий проход
/E	Печать ошибок
⏎	Запрещение вывода или при вводе - низкоскоростное перфоленточное устройство ввода

Ассемблер начинает диалог с печати на ЭПМ

ж S

что означает: "С какого устройства вводить исходный модуль?"

Ответ может быть:

H высокоскоростное перфоленточное устройство ввода;

⏎ низкоскоростное перфоленточное устройство ввода;

L низкоскоростное перфоленточное устройство ввода;

T клавиатура ЭПМ.

Ответы пользователя должны заканчиваться нажатием клавиши ⏎.

После ответа на вопрос ж S ассемблер печатает

ж B

что означает: "На какое устройство выводить объектный модуль?"

Ответ может быть:

H высокоскоростной перфоратор;

L низкоскоростной перфоратор;

⏎ запрещение вывода объектного модуля.

Ассемблер допускает изменение режима работы. Если после того как напечатано H, напечатать /3, то ассемблер будет выводить объектный модуль при третьем проходе. Если после H напечатать /E, то ассемблер будет выдавать на этом же проходе сообщения об ошибках. Если напечатаны /3 и /E, то /3 должно предшествовать /E.

Примеры:

ж B H /3/ E вывод объектного модуля на высокоскоростной перфоратор и печать ошибок на ЭПМ во время третьего прохода

ж B ⏎ вывод объектного модуля запрещается

После ответа на вопрос ж B ассемблер печатает

ж L

что означает: "На какое устройство выводить листинг?"

Ответ может быть:

T	печатающее устройство ЭВМ;
L	низкоскоростной перфоратор;
H	высокоскоростной перфоратор;
P	высокоскоростное печатающее устройство;
⇟	запрещение вывода листинга.

Ассемблер допускает изменение режима работы. Если после того как напечатано H, напечатать /2, то ассемблер будет выдавать листинг на перфоратор при втором проходе. Если после H напечатать /E, то ассемблер будет на этом же проходе выдавать на ЭВМ сообщения об ошибках. Если напечатаны /2 и /E, то /2 должно предшествовать/ E.

Примеры:

*L H/2/E	вывод листинга на высокоскоростной перфоратор и печать ошибок на ЭВМ во время второго прохода
*L ⇟	вывод листинга запрещается

После ответа на вопрос *L ассемблер печатает

*T

что означает: "На какое устройство выводить таблицу имен пользователя?"

Ответ может быть:

T	печатающее устройство ЭВМ;
H	высокоскоростной перфоратор;
L	низкоскоростной перфоратор;
P	высокоскоростное печатающее устройство;
⇟	запрещение печати таблицы имен пользователя.

Таблица имен пользователя будет выдана в конце первого прохода, если ответ на вопрос *T не будет содержать /2 или /3. Если ответ будет содержать /E, то на этом же проходе будет печать ошибок.

Перед ответом на последний вопрос начального диалога, необходимо в перфоленточное устройство ввода поместить ленту исходного модуля. Если исходный модуль приготовлен на нескольких лентах, то поместить первую ленту исходного модуля.

Таблица имен пользователя выдается на ЭВМ в алфавитном порядке. Каждое имя печатается со своим значением и соответствующим признаком. Признак указывает является ли имя меткой, регистром, относительным и/или глобальным, или оно определено оператором пря-

мого присваивания. Если имя неопределено, то вместо его значения печатается ******. В таблицу эти имена заносятся как имена, неопределенные прямым присваиванием.

Примеры:

ABCDEF		001244	метка определена
R3	=	000003	имя регистра
DIRASM	=	177777	прямое присваивание
XYZ	=	******	прямое присваивание неопределено
R6	=	******	имя регистра неопределено
LABEL	=	******	метка неопределена
SCMD		003243R	относительное имя
SDCO		005046G	глобальное имя
SECO		005076RG	относительное и глобальное имя

Если пользователю необходимо исправить ответ, то нажав клавишу ЗБ (печатается "\") можно стереть ранее напечатанный символ, а для стирания всей строки необходимо нажать комбинацию клавиш СКЛ и U (печатается " ↑U "). Это возможно, если после ответа пользователель не нажал клавишу ↵ . Если ответ пользователя не содержит символа из табл.5, то печать соответствующего вопроса повторяется.

Если в начальном диалоге не было запрещения вывода и печать ошибок не задана, а в качестве устройств ввода-вывода заданы низкоскоростные устройства, то ассемблер выполняет следующие операции:

Проход I. Ассемблер создает таблицу имен пользователя. Если в процессе трансляции встретится недопустимый оператор типа ".=выражение", то печатается этот оператор с соответствующим сообщением об ошибке. Возникшая ошибка должна быть исправлена. Неопределенные имена, если они встречаются в процессе трансляции, за исключением внешних глобальных имен, печатаются в конце прохода. Затем распечатывается таблица имен пользователя в алфавитном порядке, включая и неопределенные имена.

Проход 2. Ассемблер перфорирует объектный модуль, печатает и перфорирует количество ошибок прохода.

Проход 3. Ассемблер перфорирует или печатает листинг и количество ошибок прохода. Операторы, в которых допущены ошибки, выдаются на печать или перфорацию с соответствующими сообщениями об ошибках.

Пример начального диалога:

*S	Н	ввод исходного модуля с высокоскоростного перфоленточного устройства ввода;
*B	Н	вывод объектного модуля на высокоскоростной перфоратор;
*L	Т	печать на ЭПМ листинга во время второго прохода;
*T	Т	печать на ЭПМ таблицы имен пользователя в конце первого прохода.

I.I3. Диалог при трансляции

Во время трансляции исходного модуля ассемблер печатает на ЭПМ различные указания пользователю:

После считывания директивы .EOT ассемблер печатает:

EOF ?

и ожидает заправки следующей перфоленты исходного модуля и нажатия клавиши ⏎.

Если для вывода листинга требуется высокоскоростной перфоратор или высокоскоростное печатающее устройство, а они выключены, то ассемблер печатает:

EOM ?

и ожидает включения соответствующего устройства и нажатия клавиши ⏎ на клавиатуре ЭПМ.

Если для вывода объектного модуля необходим высокоскоростной перфоратор, а он выключен, то ассемблер печатает:

EOM ?
*S

и трансляция прекращается. Повторяется начальный диалог.

После считывания директивы .END , ассемблер печатает:

END ?

Пользователь должен поместить в устройство ввода первую ленту исходного модуля и нажать клавишу ⏎ для выполнения следующего прохода. При последнем проходе директива .END вызывает начальный диалог.

Если для получения объектной ленты включается низкоскоростной перфоратор до нажатия клавиши ⏎ , то на ленте пробиваются коды символов ВК и ПС, но Абсолютным загрузчиком они игнорируются.

Если последняя лента заканчивается директивой .EOT, то директиву .END можно получить, если в ответ на EOF ? нажать

клавиши E и ⏎ . Далее ассемблер печатает:

END?

и ожидает нажатия клавиши ⏎ для выполнения следующего прохода

Пример:

EOF ? E ⏎

END?

В любой момент трансляции пользователь может повторно запустить ассемблер и восстановить начальный диалог с помощью нажатия клавиш СКЛ и Р (клавиша "Р" на нижнем регистре клавиатуры ЭПМ).

I.I4. Печать листинга

Листинг печатается по страницам. В начале каждой страницы указывается ее восьмеричный номер. Директивы .ASECT и .CSECT вызывают печать текущего значения счетчика адресов (абсолютное или относительное).

Формат печати листинга следующий:

```
EELLLLLL  OOOOOOASSS...S
          OOOOOO
          OOOOOO
```

где E - поле ошибки;
L - поле адреса;
O - поле команды;
A - апостроф;
S - поле оператора;

Поле ошибки содержит сообщение об ошибке. Поле адреса содержит восьмеричный адрес команды. Поле команды содержит объектные данные в восьмеричной системе. Данные, в зависимости от типа команды и метода адресации, занимают от одного до трех машинных слов. Второе и третье слово располагается под первым командным словом, а т.к. порядок расположения адресов последовательный, им адреса не предшествуют. Если апостроф присутствует, то второе слово команды или третье, или оба будут модифицироваться программой "Компоновщик". Поле оператора содержит оператор исходной программы (не более 72_{10} знаков).

I.I5. Вывод объектного модуля

Перемещающий Ассемблер выдает на объектную ленту следующие данные:

а) таблицу определений глобальных имен;
б) блоки перемещений;
в) блоки текста.

Таблица определения глобальных имен формируется в начале второго прохода, а блоки перемещений и блоки текста формируются в процессе трансляции. Объектный модуль, созданный Перемещающим ассемблером, обрабатывается только программой "Компоновщик". На объектном модуле начало каждого блока указывается кодом 001. Компоновщик игнорирует всю информацию, находящуюся на ленте перед кодом 001.

Таблица определения глобальных имен содержит данные о глобальных именах (точках входа и внешних именах), которые определены в данном модуле и могут быть использованы в других модулях. Каждому глобальному имени в таблице соответствует блок, содержащий это имя в коде RAD50, признак имени (абсолютный или относительный) и значение имени (для наименования секции - размер секции).

Данные таблицы определения глобальных имен необходимы Компоновщику для определения абсолютных адресов глобальных имен.

Блоки перемещений состоят из двоично-символических данных. С помощью этих данных Компоновщик перемещает и связывает блоки текста.

Блоки текста состоят полностью из машинных данных. Операнды находятся в немодифицированном виде. Вывод блока текста происходит в том случае, когда буфер блока текста или блок перемещений заполнены, либо когда счетчик адресов требует модификации.

I.16. Печать ошибок

С помощью начального диалога может быть задана печать ошибок при любом проходе. Печать ошибок не имеет специального формата: печатается оператор (см.п.I.14.), в котором поле ошибки содержит не более двух сообщений об ошибке.

Типы возможных ошибок перечислены в табл.6.

Таблица 6

Сообщение об ошибке	Значение
A	Ошибка адресации. В операторе использована неправильная адресация
B	Оператор транслируется по нечетному адресу. Содержимое счетчика адресов увеличивается на 1
D	Обращение к имени, которое определено более одного раза
I	Недопустимый символ. При распечатке недопустимый символ заменяется символом "?"
L	Переполнение буфера строки. Все символы, стоящие после первых 72 символов, игнорируются
M	Многократное определение метки. Метка эквивалентна по первым шести символам ранее встреченной метке
N	Десятичное число не заканчивается десятичной точкой
P	Значение метки меняется от прохода к проходу
Q	Сомнительный синтаксис. Пропущен аргумент, не закончен просмотр оператора или за символом ВК не следует символ ПС или ПФ
R	Ошибка регистра. Обращение к несуществующему регистру
S	Переполнение таблицы имен. Имя пользователя вызывает переполнение таблицы имен ассемблера. Трансляция прекращается. Повторяется начальный диалог
T	Ошибка усечения. Сформированное число занимает более 16 значащих разрядов или результат выражения, обрабатываемого директивой .BYTE, имеет более 8 разрядов
U	Имя неопределено. Неопределенному имени присваивается нулевое значение.

I.I7. Остановы

Если возникает непредусмотренное прерывание, то происходит останов (см.табл.7).

Таблица 7

Адрес останова	Причина останова
12	Прерывание по резервной команде
16	Прерывание по Т-разряду
26	Прерывание по нарушению питания
32	Прерывание по команде EMT
36	Прерывание по команде TRAP
40	Ошибка, обнаруженная программой "Супервизор ввода-вывода".

2. АССЕМБЛЕР

Язык Ассемблер соответствует языку Перемещающего ассемблера, но в отличие от него является абсолютным языком, т.к. в нем отсутствуют средства по созданию перемещаемых объектных модулей: метка, переменная, выражение и счетчик адреса принимают только абсолютные значения. Результатом трансляции является модуль в абсолютном формате (см. п.6.2).

Кроме того, в языке Ассемблер отсутствуют директивы

- присвоения наименования объектному модулю

  .TITLE

- границы программы

  .LIMIT

- преобразования символов

  . RAD 50

- секций

  .ASECT

  .CSECT

- условной трансляции

  .IFZ

  .IFNZ

  .IFLE

  .IFG

  .IFGE

  .IFDF

  .IFNDF

- окончания

  .ENDC

и команды

SXT, MFPS , MTPS , XOR, MUL , DIV , SOB , RTT, ASH, ASHC, MARK

ВНИМАНИЕ!

Страницы с 41 по 67 заменены настоящей страницей.

3. КОМПОНОВЩИК

Программа "Компоновщик" предназначена для связывания и перемещения объектных модулей (см.п.3.3) и получения загрузочного модуля (см.п.3.6.).

Обрабатывая объектные модули, Компоновщик осуществляет:

а) перемещение каждого объектного модуля и присвоение абсолютных адресов;

б) связывание модулей с помощью глобальных имен;

в) печать карты загрузки, указывающей распределение памяти;

г) перфорацию загрузочного модуля.

Использование Компоновщика дает ряд преимуществ.

Программу на исходном языке можно разбивать на модули (обычно подпрограммы) и транслировать их отдельно. Если в модуле обнаружена ошибка, то вновь транслируется только этот модуль. Компоновщик может затем связать только что полученный объектный модуль с остальными модулями.

Нет необходимости присваивать абсолютные адреса с помощью оператора прямого присваивания: Компоновщик присваивает их автоматически. Такое присвоение адресов не допускает наложения подпрограмм в памяти.

Помодульная трансляция позволяет также использовать общее количество имен в программе, превышающее количество допустимых имен в одном модуле, причем внутренние имена в различных модулях могут быть одинаковыми.

3.1. Абсолютная и перемещаемая секции программы

Программа, транслируемая с помощью Перемещающего ассемблера, может состоять из абсолютной секции, определяемой директивой .ASECT, и относительной секции, определяемой директивой .CSECT. Если программа не содержит ни одной из указанных директив, то Перемещающий ассемблер предполагает наличие директивы .CSECT.

В абсолютной секции абсолютные адреса присваиваются с помощью оператора прямого присваивания (см.п.1.5).

В относительной секции абсолютные адреса присваиваются Компоновщиком так, чтобы программа размещалась в старших адресах памяти. Все команды Компоновщик модифицирует последовательно,

вычисляя адреса для перемещения относительных (управляемых) секций.

Именованные управляемые секции обрабатываются Компоновщиком глобально, то есть управляемым секциям с одинаковыми наименованиями, содержащимся в разных объектных модулях, Компоновщик присваивает один и тот же абсолютный адрес; длина секции выбирается по максимальной для всех секций. Таким образом, присвоение наименований позволяет выделить подпрограммы и/или данные для совместного использования в различных объектных модулях. Компоновщик не допускает одинаковых с глобальными именами наименований управляемых секций. Это вызывает ошибку повторного определения имени.

Абсолютные секции имеют наименование .ABS и обрабатываются аналогично именованным управляемым секциям, сохраняя присвоенные абсолютные адреса.

3.2. Глобальные имена

Глобальные имена обеспечивают связь объектных модулей.

Глобальные имена описываются директивой .GLOBL.

Глобальное имя, определенное в модуле как метка или прямым присваиванием, является именем входа (точкой входа) модуля. Обращение к такому имени может содержаться в любом другом модуле.

Глобальное имя, не определенное в модуле, является внешним именем модуля. Это имя должно быть определено в каком-либо другом модуле.

Имя, не описанное директивой .GLOBL, является внутренним именем модуля. Внутреннее имя может использоваться только в данном модуле.

Компоновщик формирует для каждого глобального имени блок данных, характеризующий это имя и обращение к нему. Затем модифицирует команды и/или данные, использующие глобальные имена.

3.3. Объектный модуль

Исходными данными для работы Компоновщика являются объектные модули, полученные с помощью Перемещающего ассемблера (см.раздел I).

3.4. Компоновка

Компоновщик формирует загрузочный модуль за два прохода.

При первом проходе Компоновщик извлекает из объектных модулей глобальные имена, содержащиеся в таблицах определения глобальных имен и присваивает абсолютные значения относительным секциям и глобальным именам, а также выдает на ЭПМ список неопределенных имен и карту загрузки, если они были заданы в диалоге. Компоновщик перемещает относительную секцию программы в старшие адреса памяти емкостью 16К байт, располагая ее перед Абсолютным загрузчиком.

Старший адрес загрузки относительной секции вычисляется следующим образом:

Старший адрес загрузки = верхний предел памяти +1 - длина Абсолютного загрузчика и Программы начальной загрузки в байтах.

Вычисленный адрес загрузки заносится в ячейку R.HLM2.

Например, при использовании памяти емкостью 16К байт

R.HLM2 = 16384-112=16272=37460_8.

Пользователь может изменить старший адрес загрузки относительной секции, используя диалог (см.п.3.9).

При втором проходе Компоновщик формирует и одновременно перфорирует загрузочный модуль, используя для этого результаты первого прохода и двоичные данные объектного модуля

3.5. Карта загрузки

Карта загрузки печатается Компоновщиком при первом проходе и отражает структуру загрузочного модуля.

Карта загрузки начинается с печати пускового адреса, нижнего и верхнего пределов (младшего и старшего адресов) относительной секции модуля. Затем печатаются данные о распределении памяти для каждого отдельного модуля: наименование модуля, наименования секций, адрес загрузки и длина секций в байтах, имена входа в алфавитном порядке с адресами.

После печати описанной выше информации перечисляются неопределенные имена в алфавитном порядке.

Заметим, что если модули A, B, C загружаются в память с помощью Компоновщика последовательно, то в памяти A будет находиться в младших адресах памяти, а C - в старших.

Ниже приводится пример печати карты загрузки.

КАРТА

ПУСКОВОЙ АДРЕС 037434

```
НИЖ  ПРЕДЕЛ  037406
ВЕРХ ПРЕДЕЛ  037460
**********
МОДУЛЬ  MOD1
СЕКЦИЯ   ВХОД       АДРЕС   ДЛИНА
<.ABS>              000000  000000
<    >              037406  000044
         X3         037452
         X4         037440
         X5         037450
         X7         037430
**********
МОДУЛЬ          MOD2
СЕКЦИЯ          ВХОД    АДРЕС       ДЛИНА
<    >                  037452      000006
                X1      037452
                X2      037452
**********
**********
НЕОПРЕДЕЛЕНЫ  ИМЕНА
X6
```

3.6. Загрузочный модуль

Результатом работы Компоновщика является загрузочный модуль-программа, отперфорированная в абсолютном формате, которую можно ввести в память ЭВМ с помощью Абсолютного загрузчика (см.п.6.2) и выполнить.

Первым блоком загрузочного модуля (24_8 байт) является блок связей. Он имеет младший (абсолютный) адрес загрузки, равный младшему перемещенному адресу программы. Последующие блоки содержат двоичные данные программы, которые при загрузке перекрывают блок связей.

Существует два случая, когда перекрытие блока связей невозможно:

а) объектные модули, используемые для образования загрузочного модуля, не имеют относительной секции программы;

б) длина относительной секции программы меньше длины блока связей.

В первом случае Компоновщик формирует адрес загрузки блока связей таким образом, чтобы адрес конца загрузки блока связей совпадал со старшим адресом загрузки относительной секции (см.п.3.4). Во втором случае Компоновщик формирует адрес загрузки блока связей равным 0.

Последним блоком загрузочного модуля является блок из шести байтов с нечетным или четным (пусковым) адресом загрузки.

При компоновке нескольких модулей с различными пусковыми адресами в качестве пускового адреса загрузочного модуля берется первый четный адрес объектного модуля. Например, при компоновке четырех модулей, имеющих на исходной ленте блоки .END, .END, .ENDA .ENDB и вводимых в указанной последовательности, пусковым адресом будет адрес A.

3.7. Загрузка и запуск

Перфолента "Компоновщик" загружается в память ЭВМ с помощью Абсолютного загрузчика в нормальном режиме (см.п.6.2) и автоматически запускается.

3.8. Повторный запуск

Повторный запуск осуществляется нажатием комбинацией клавиш СКЛ и Р (буква "р" на нижнем регистре клавиатуры ЭПМ) во время работы Компоновщика.

Нажатие клавиш СКЛ и Р во время печати карты загрузки прекращает печать карты загрузки и осуществляет переход к диалогу.

3.9. Диалог

Диалог служит для определения устройств ввода-вывода и управления процессом компоновки.

После начального или повторного запуска программа "Компоновщик" печатает наименование программы и вопрос для определения устройства ввода в виде:

ж I

Возможен один из следующих ответов:

┘ или Н┘ – считывание объектных модулей с высокоскоростного перфоленточного устройства ввода;

└┘ – считывание объектных модулей с низкоскоростного перфоленточного устройства ввода.

Вопрос для определения перфоратора печатается в виде:

ж О

Возможные ответы:

↵ или Н↵ - вывод загрузочного модуля на высокоскоростной перфоратор;

L↵ - вывод загрузочного модуля на низкоскоростной перфоратор

Вопрос о выводе карты загрузки печатается в виде:

ж М

Возможные ответы:

↵ - запрещение вывода;

Т↵ - вывод на печатающее устройство ЭПМ;

Н↵ - вывод на высокоскоростной перфоратор;

Р↵ - вывод на высокоскоростное печатающее устройство.

Вопрос об определении старшего адреса загрузки относительной секции печатается в виде:

ж Т

Возможные ответы:

↵ - старший адрес загрузки не изменяется (первоначально установлен на 37460);

N↵ - старший адрес загрузки равен N.

Компоновщик по заданному старшему адресу и длине программы пользователя, содержащейся в объектных модулях, вычисляет младший адрес загрузки относительной секции.

Если ответ на вопрос "ж Т" был "↵", то печатается вопрос об определении младшего адреса загрузки относительной секции в виде:

ж В

Возможные ответы:

↵ - младший адрес загрузки вычисляется по старшему адресу;

N↵ - младший адрес загрузки равен N.

Компоновщик по заданному младшему адресу N и длине программы пользователя, содержащейся в объектных модулях, вычисляет и переустанавливает старший адрес загрузки относительной секции.

Первый проход работы Компоновщика начинается с печати сообщения:

ПРОХ 1

Печать "*" означает запрос на считывание объектного модуля:
Возможные ответы:

↵ – считывание объектного модуля и запрос на считывание следующего модуля;

U↵ – печать неопределенных глобальных имен (если они есть) и запрос на считывание следующего объектного модуля;

E↵ – печать неопределенных глобальных имен (если они есть) и запрос на считывание следующего объектного модуля. Если нет неопределенных глобальных имен, печать карты загрузки (если она была затребована в начале диалога) и переход ко второму проходу;

C↵ – конец считывания. Присвоение нулевых значений всем неопределенным глобальным именам, печать карты загрузки (если она была затребована в начале диалога) и переход ко второму проходу.

Второй проход работы Компоновщика начинается с печати сообщения:

ПРОХ 2

и перфорации синхродорожки и блока связей.

После этого печатается запрос на считывание объектного модуля:

*

Возможен один ответ:

↵ – считывание очередного модуля и запрос на считывание следующего модуля.

Перед печатью ↵ нужно включить перфоратор.

Последовательность, в которой считывались модули при первом проходе, должна быть сохранена при втором проходе.

После считывания последнего модуля Компоновщик заканчивает перфорацию загрузочного модуля и передает управление на начало диалога.

В начале и в конце загрузочного модуля перфорируется примерно по 40 см синхродорожки.

Если для вывода загрузочного модуля используется низкоскоростной перфоратор, то он должен быть включен до начала второго

прохода, т.е. перед печатью E↵ или C↵ . Коды этих знаков и карта загрузки выдаются на перфоленту. Эта информация отделяется от двоичной синхродорожкой и при считывании загрузочного модуля по Абсолютному загрузчику игнорируется. Можно избежать перфорации данной информации, включив перфоратор сразу же после печати ПРОХ 2.

Примечание. При распознавании ответов на все вопросы диалога, кроме "ж Т"и"ж В", анализируется код, предшествующий возврату каретки, поэтому

A B C D E F G H↵

эквивалентно H↵

3.IO. Ошибки

Ошибки, обнаруженные Компоновщиком, делятся на устранимые и неустранимые.

При возникновении устранимой ошибки печатается диагностическое сообщение, Компоновщик принимает меры по устранению ошибки и продолжает работу.

При возникновении неустранимой ошибки печатается диагностическое сообщение, и Компоновщик переходит к диалогу (см.п.3.9.).

3.10.I. Устранимые ошибки

К устранимым ошибкам относятся следующие:

а) ? ИМЯ МОДУЛЯ НЕЕДИНСТВ

Наименование объектного модуля неединственно. Модуль игнорируется.

Компоновщик запрашивает считывание следующего объектного модуля (печатается "ж").

б) ? КАРТА КН

НАЖАТЬ ВК ДЛЯ ПРОД

Устройство, выбранное для печати карты загрузки, выключено.

Компоновщик для продолжения работы требует включения устройства и нажатия клавиши ↵ на клавиатуре ЭПМ.

в) ? ОШИБКА ПЕРЕМЕЩЕНИЯ - АБС АДРЕС XXXXXX

Ошибка определяется по ненулевому содержимому старшего байта перемещенного (или связанного) значения. Значение усекается до 8 разрядов.

г) ? ЗАГРУЗИТЬ XXXXXX

Нарушен порядок загрузки модулей при втором проходе.

В сообщении печатается наименование модуля, который должен быть загружен.

д) ? XXXXXX ПОВТОРНО ОПРЕДЕЛЕН МОДУЛЕ XXXXXX

Повторное определение глобального имени. Повторное определение игнорируется.

3.I0.2. Неустранимые ошибки

К неустранимым ошибкам относятся следующие:

а) ? ПЕРЕПОЛНЕНА ТАБЛ ИМЕН МОДУЛЬ XXXXXX ИМЯ XXXXXX

Переполнение таблицы определения глобальных имен.

б) ОШИБКА СИСТЕМЫ XX,

где XX – код ошибки.

Коды и причины этих ошибок приведены в табл.I3.

Таблица I3

Код ошибки	Причина ошибки
01	Вводимое имя не распознаваемо
02	Блок перемещения содержит глобальное имя, которого нет в таблице определения глобальных имен
03	Ячейка модификации счетчика адресов, содержащаяся в блоке перемещения, не является последней
04	Объектный модуль начинается не с таблицы определения глобальных имен
05	Первое имя в таблице определения глобальных имен не является наименованием модуля
06	Блок перемещений содержит наименование необнаруженной секции
07	Пусковой адрес указывает на несуществующее наименование модуля
08	Пусковой адрес указывает на несуществующее имя
09	Индекс таблицы внутренних переходов вышел за диапазон
10	Несовпадение контрольных сумм при считывании объектного модуля
11	Блок текста объектного модуля больше допустимого (64_{10} слова)
12	Ошибка устройства вывода

Компоновщик не выдает сообщение об ошибке при загрузке программы пользователя в память ниже 0-го адреса. Чтобы избежать эту ошибку, необходимо проверить нижний и верхний пределы памяти, печатающиеся в карте загрузки.

3.II. Остановы

Если возникает непредусмотренное прерывание, то происходит останов (см.табл.I4). По адресу останова можно определить причину останова.

Таблица I4

Адрес останова	Причина останова
12	Прерывание по резервной команде
16	Прерывание по Т-разряду
26	Прерывание по нарушению питания
32	Прерывание по команде EMT
36	Прерывание по команде TRAP
40	Ошибка, обнаруженная программой "Супервизор ввода-вывода"

4. РЕДАКТОР ТЕКСТА

4.I. Назначение

Программа "Редактор текста" предназначена для редактирования исходного текста (программы), внесения в него изменений и дополнений, и вывода всего или части текста на перфорацию или печать. Редактирование исходного текста происходит в режиме диалога пользователя с ЭВМ.

4.2. Определения

Страницей текста назовем текст, ограниченный знаком ПФ.

Буфером страниц назовем область памяти, предназначенную для хранения текста. Размер буфера страниц зависит от емкости используемой памяти (см.п.4.4.2). Исходный текст загружается в буфер страниц с клавиатуры ЭПМ или перфоленточного устройства ввода. В буфер страниц можно загрузить более одной страницы текста.

"Точкой" назовем указатель первого необработанного знака текста.

Маркером назовем указатель знака, используемый для запоминания положения "точки".

Таким образом, "точка" находится между последним обработанным знаком исходного текста и следующим знаком текста и означает "здесь", маркер остается на месте при перемещении "точки" и означает "там". Установка "точки" и маркера осуществляется по командам редактора.

4.3. Команды и функции программы "Редактор текста"

Входной информацией для программы "Редактор текста" является исходный текст и команды пользователя, подаваемые с клавиатуры ЭПМ. Пользователь может подать очередную команду после окончания начального диалога (см.п.4.I0) или выполнения текущей команды, о чем свидетельствует печать на ЭПМ знака " * ".

Программа "Редактор текста" работает в двух режимах: командном и текстовом. В командном режиме подаваемые данные интерпретируются как команды, в текстовом – как обрабатываемый текст. Все команды редактора, в том числе команды, предназначен-

ные для обработки текста, подаются в командном режиме. Первоначально (после окончания начального диалога) программа "Редактор текста" находится в командном режиме.

Команды редактора по их назначению можно разделить на следующие группы:

а) команды ввода-вывода ("Открыть","Чтение","Печать","Перфорация", "Следующий","Перевод формата", "Конец ленты");

б) команды перемещения точки и маркера ("Начало","Конец","Переход", "Продвижение","Маркер");

в) команды поиска ("Поиск","Поиск общий");

г) команды редактирования текста ("Вставка","Стирание знаков", "Стирание строк","Замена знаков","Замена строк").

Перечень команд программы "Редактор текста" приведен в Приложении 3.

Некоторые команды редактора требуют специального аргумента. Аргумент используется для уточнения текста, с которым будет работать команда, или определяет число раз выполнения команды. Команды редактора используют следующие аргументы $\pm n$, "0", " @ ","/".

Аргумент n может быть любым положительным или отрицательным числом от 1 до 32767 ($2^{15}-1$). Если n не содержит знака, то аргумент считается положительным. В тех командах, где аргумент необходим, но он отсутствует - полагается равным "1" (или "-1" при наличии знака "-").

Аргумент "0" отсылает к началу текущей строки.

Аргумент "@" отсылает к маркеру.

Аргумент "/" отсылает к концу буфера страниц

Строкой будем называть последовательность знаков, ограниченную знаком ПС, ПФ или ВК.

По возврату каретки программа "Редактор текста" выполняет перевод строки. Строку, содержащую "точку", назовем текущей строкой.

Команда является строчно-ориентированной, если она оперирует со строками. В строчно-ориентированных командах аргумент указывает на число строк.

Команда является знако-ориентированной, если она оперирует со знаками. В знако-ориентированных командах аргумент указывает

на число знаков.

Все команды редактора подаются на нижнем регистре клавиатуры ЭПМ и заканчиваются нажатием клавиши ⏎. Нажатие клавиши ⏎ переводит программу "Редактор текста" в текстовый режим при выполнении команд "Вставка", "Замена знаков", "Замена строк", "Поиск общий", при этом можно включать в текст непечатный знак ГТ (клавиши СКЛ и I), по которому выполняется функция табулирования при выводе на печать текста (дополнение пробелами до 8 числа напечатанных знаков). Нажатие клавиши ↑ переводит программу "Редактор текста" из текстового режима в командный режим.

При печати несуществующей команды или нарушении формата команды, при перевыполнении буфера страниц или задании аргумента, отсылающего за пределы буфера страниц, программа "Редактор текста" печатает "?" и на следующей строке - "*". Затем "Редактор текста" входит в командный режим и ждет дальнейших команд.

В дальнейшем, то, что печатает программа "Редактор текста", подчеркнуто, а положение точки указано знаком "↑".

4.4. Команды ввода-вывода

4.4.I. Команда "Открыть"

Для выполнения команды напечатать:

О⏎

Команда "Открыть" используется в тех случаях, когда редактируемый текст находится на нескольких перфолентах и для заправки очередной ленты устройство выключается. После включения устройства должна быть подана команда "Открыть" и команда ввода.

Эта команда аналогична команде "Поиск" программы "Супервизор ввода-вывода".

4.4.2. Команда "Чтение"

Для выполнения команды напечатать:

К⏎

Команда "Чтение" используется для загрузки страницы текста с перфоленточного устройства ввода. Загружаемый текст при этом добавляется к содержимому буфера страниц и записывается за последней строкой.

Текст будет загружаться:

- до появления знака ПФ;

- пока до заполнения буфера страниц не останется 128 знаков;
- до появления знака ПС и когда до заполнения буфера осталось не более 500 знаков;
- до выключения устройства ввода или физического конца перфоленты.

ЭВМ с оперативной памятью емкостью 16К байт позволяет ввести в буфер страниц до 12000 знаков (буфер страниц начинается с ячейки 010414). Каждые добавляемые 8К байт памяти позволяют дополнительно ввести 8000 знаков.

После выполнения команды "Чтение" "точка" и маркер будут расположены в начале буфера страниц.

4.4.3. Команды "Печать" и "Перфорация"

Для выполнения команды напечатать:

<аргумент> L ⏎ ("Печать")

или

<аргумент> P ⏎ ("Перфорация")

Команда "Печать" ("Перфорация") используется для вывода на печать (перфорацию) текста, количество которого указано в аргументе:

nL – печать nP – перфорация	знаков, начиная с "точки" и кончая n-ой строкой
nL – печать nP – перфорация	знаков, начиная с первого знака n-ой предыдущей строки и кончая "точкой"
oL – печать oP – перфорация	знаков, начиная с первого знака текущей строки и кончая "точкой"
@L – печать @P – перфорация	знаков между "точкой" и маркером
/L – печать /P – перфорация	знаков, начиная с "точки" и кончая последним знаком буфера страниц.

В дополнение к вышеуказанным командам печати имеются три специальные команды печати, для выполнения которых не требуется аргумент.

V ⏎ распечатывает текущую строку, т.е. строку, содержащую "точку"

<\J аналогична команде -1L (если "точка" расположена в начале строки, то команда <\J распечатывает строку, предшествующую текущей строке).

>\J распечатывает строку, следующую за текущей строкой.

Команда "Печать" ("Перфорация") не вызывает перемещения "точки".

Примеры:

Исходный текст	Команда	Результат
BEQ CK	ж -L\J	BEQ CK
BR PUT		BR P
BEQ ЖОК	ж L\J	Q ЖОК
BNE ОКО	ж >\J	TST EKOCNT
TSTB EKOCNT		
CMPB ICHAR,#33	V	CMP ICHAR,#175
BEQ ЖALT	<	BEQ ЖALT
CMPB ICHAR,#175		CMPB I
BNE PLACE	>	BNE PLACE

4.4.4. Команда "Следующий"

Для выполнения команды напечатать: nN\J

Команда используется для выполнения операций в следующей последовательности:

- вывода на перфорацию содержимого буфера страниц и зоны синхродорожки (≈ 10 см), если последним знаком буфера страниц является знак ПФ. Первоначально буфер страниц независимо от числа содержащихся в нем страниц текста перфорируется за один раз;
- стирания содержимого буфера страниц;
- загрузки следующей страницы текста.

Эта последовательность выполняется n раз.

Если n превышает число страниц исходной ленты с учетом первоначального содержимого буфера страниц, то описанная последовательность операций выполняется имеющееся число раз и затем печатается "?".

После выполнения команды "точка" и маркер будут находиться в начале буфера страниц.

4.4.5. Команда "Перевод формата"

Для выполнения команды напечатать:

F⏎

Команда используется для перфорации кода знака ПФ (14) и зоны синхродорожки (≈ 10 см)

4.4.6. Команда "Конец ленты"

Для выполнения команды напечатать:

nT⏎

Команда используется для перфорации n десятисантиметровых зон синхродорожки.

4.4.7. Использование низкоскоростного перфоратора

Перед выполнением команд "Перфорация", "Перевод формата", "Конец ленты", "Следующий", "Поиск общий" программа "Редактор текста" циклится, ожидая включения перфоратора. После включения перфоратора печать пробела приводит к выполнению команды. После выполнения команды редактор вновь циклится, ожидая выключения перфоратора. После выключения перфоратора печать пробела возвращает редактор в командный режим.

4.5. Команды перемещения "точки" и маркера

4.5.1. Команды "Начало" и "Конец"

Для выполнения команды напечатать:

B⏎ ("Начало")

или

E⏎ ("Конец")

Команда "Начало" используется для перемещения "точки" в начало буфера страниц, команда "Конец" – в конец буфера страниц.

4.5.2. Команды "Переход" и "Продвижение"

Для выполнения команды напечатать:

<аргумент> J⏎ ("Переход")

или

<аргумент> A⏎ ("Продвижение")

Команды "Переход" и "Продвижение" используются для перемещения "точки" соответственно на число знаков или строк, определяемое аргументом.

*n*J	перемещает "точку" вперед на *n* знаков
*n*A	перемещает "точку" вперед к каналу n-й строки
-*n*J	перемещает "точку" назад на *n* знаков
-*n*A	перемещает "точку" назад к каналу n-ой предыдущей строки
0J или 0A	перемещает "точку" в начало текущей строки
@J или @A	перемещает "точку" к маркеру
/J или /A	перемещает "точку" в конец буфера страниц.

4.5.3. Команда "Маркер"

Для выполнения команды напечатать:
M⏎

Эта команда используется для установки маркера, т.е. для запоминания положения "точки", обращение к которой осуществляется по командам с аргументом @. Запоминается только одно положение "точки".

4.6. Команды поиска

4.6.I. Команда "Поиск"

Для выполнения команды напечатать:
*n*G⏎

После этого напечатать строку текста, не превышающую 42-х знаков, n-ое появление которой необходимо найти. Текстовый режим завершается нажатием клавиши ⏎.

По команде *n*G⏎ осуществляется поиск строки текста в буфере страниц, начиная с "точки". После нахождения n-го появления строки текста, она распечатывается (выполняется команда 0L).

После выполнения команды "точка" будет находиться за найденным текстом или, если текст не найден, - в конце буфера страниц.

Редактор печатает "?" в следующих случаях:
- длина разыскиваемого текста > 42 знаков;
- разыскиваемый текст не найден.

Примеры:

Исходный текст	Команда	Результат
BEQ ¤ALT	* 2G⏎	BNE ¤ALT
CMPB ICHAR, #175	¤ALT	
BNE ¤ALT		
CMPB ICHAR, #RUBOUT	* G⏎	BR
BEQ SITE	TE⏎	
BR PUT	BR⏎	

4.6.2. Команда "Поиск общий"

Для выполнения команды напечатать:
H⏎

После этого напечатать строку текста не превышающую 42-х знаков которую необходимо найти. Текстовый режим завершается нажатием клавиши ⏎.

Команда "Поиск общий" включает в себя команды "Поиск" и "Следующий", осуществляя поиск текста, начиная с "точки" и просматривая весь текст до нахождения искомого. Если текст не найден в буфере страниц, то на ленте перфорируется содержимое буфера, затем буфер очищается и записывается новая страница. Это будет продолжаться до тех пор, пока не будет найден искомый текст. Найденный текст распечатывается (выполняется команда OL). Если текст не найден, редактор печатает "?".

После выполнения команды "точка" будет находиться за найденным текстом, маркер- в начале той страницы, где найден текст, или, если текст не найден, "точка" будет находиться в конце буфера страниц.

Команда "Поиск общий" при отсутствии искомого текста применяется для отмены редактирования, т.е. весь оставшийся текст вводится с исходной ленты и выводится на перфоленту.

4.7. Команды редактирования

4.7.1. Команда "Вставка"

Для выполнения команды напечатать:

I ⏎

Затем напечатать вставляемый текст построчно (не более 80 знаков в строке). Текстовый режим завершается нажатием клавиши ⏎.

По команде I ⏎ печатаемый текст записывается в буфер страниц, начиная с "точки".

После выполнения команды "точка" находится за последним вставленным знаком текста.

Примеры:

Исходный текст	Команда	Результат
INC R↓N	* I⏎ DU⏎	INC RDU↓N
BEQ CK↓BR PUT	* I⏎ ⏎⏎	BEQ CK ↓BR PUT

Команда "Вставка" используется для получения исходной перфоленты.

4.7.2. Команды "Стирание знаков" и "Стирание строк"

Для выполнения команды напечатать:

<аргумент> D⏎ ("Стирание знаков")

или

<аргумент> K⏎ ("Стирание строк")

Команды "Стирание знаков" и "Стирание строк" используются для стирания соответственно знаков и строк, количество которых указывается в аргументе (см. табл. I5).

После выполнения команд с аргументом "0", "@", "/" маркер находится на месте "точки", а с аргументом *n* – за последним стертым знаком.

4.7.3. Команды "Замена знаков" и "Замена строк"

Для выполнения команды напечатать:

<аргумент> C⏎ ("Замена знаков")

или

<аргумент> X⏎ ("Замена строк")

Затем напечатать текст для замены. Текстовый режим завершается нажатием клавиши ⏎ .

Команды "Замена знаков" и "Замена строк" используются для замены в тексте соответственно знаков и строк, количество которых указывается в аргументе (см.табл.I5).

Если в качестве аргумента используется ±n, то он интерпретируется так же, как и в команде "Стирание знаков" или "Стирание строк".

Примеры:

Исходный текст	Команда	Результат
; ПЕРЕЙТИ, ЕСЛИ НЕТ↓	* -3C⏎	;ПЕРЕЙТИ,ЕСЛИ ДА
	ДА ⏎	
DEC * R5	* X⏎	DEC * R5
↑JMP ОКО	BR ОКО ⏎	BR ОКО↓

После выполнения команд "точка" располагается после введенного текста. После выполнения команд с аргументом "0", " @ ", "/" маркер находится на месте "точки".

Таблица I5

Команды стирания и замены

Знако-ориентированные команды		Строчно-ориентированные команды	
nD стирает nC заменяет	n последующих знаков	nK стирает nX заменяет	n знаки строк, начиная с "точки" и заканчивая n-ой строкой включительно
-nD стирает -nC заменяет	n предыдущих знаков	-nK стирает -nX заменяет	n знаки строк, начиная с n-ой предыдущей строки и кончая "точкой"
0D стирает 0C заменяет	текущую строку до "точки"	0K стирает 0X заменяет	текущую строку до "точки"

Продолжение табл. I5

Знако-ориентированные команды	Строчно-ориентированные команды
@D стирает, @C заменяет } знаки строк, заключенных между "точкой" и маркером	@K стирает, @X заменяет } знаки строк, заключенных между "точкой" и маркером
/D стирает, /C заменяет } знаки строк, начиная с "точки" и кончая последним знаком буфера страниц	/K стирает, /X заменяет } знаки строк, начиная с "точки" и кончая последним знаком буфера страниц

Если введена часть текста и происходит переполнение буфера, то данные последней введенной строки пропадут, но стирания всего текста не произойдет. Переполнения буфера можно избежать, если последовательно выполнить команды "Стирание знаков" или "Стирание строк" и "Вставка".

4.8. Загрузка и запуск

Программа "Редактор текста" загружается в память ЭВМ с помощью Абсолютного загрузчика в нормальном режиме (см.п.6.2.) и автоматически запускается.

4.9. Повторный запуск

Для повторного запуска программы "Редактор текста" в любой момент времени необходимо нажать 2 раза комбинацию клавиш СКП и Р (буква "Р" на нижнем регистре клавиатуры ЭПМ) (печатается " _Р", " ж ", " _ Р"). После этого редактор печатает "ж".

Адресом повторного пуска программы "Редактор текста" является адрес 003746.

4.IO. Начальный диалог

Программа "Редактор текста" переходит к начальному диалогу после начального или повторного запуска.

Начальный диалог служит для определения перфоленточных устройств ввода-вывода.

Первый вопрос, печатаемый редактором,

ж 1

означает: "Какое перфоленточное устройство ввода?"

Возможные ответы"

H ⏎ - высокоскоростное

L ⏎ - низкоскоростное

Второй вопрос

ж 0

означает: "Какое перфоленточное устройство вывода?"

Возможные ответы:

H ⏎ - высокоскоростное

L ⏎ - низкоскоростное

При использовании низкоскоростного перфоленточного устройства вывода редактор печатает вопрос

НПР ВЫКЛ ?

который позволяет проверить выключение устройства.

Затем необходимо нажать клавишу ⏎. Редактор печатает "ж", что указывает на готовность редактора к приему команд.

При использовании высокоскоростного перфоленточного устройства вывода редактор сразу после ответа на второй вопрос печатает " ж ".

4.II. Исправление ошибок

При редактировании текста бывает необходимо исправить ошибки в подаваемых командах. Для этого имеются четыре специальные команды:

а) нажатие клавиши ЗБ (печатается " \ ") стирает предыдущий знак текущей строки;

б) нажатие комбинации клавиш СКЛ и U (печатается "←U.") стирает все знаки текущей строки;

в) нажатие комбинации клавиш СКЛ и Р (печатается " ←Р", "ж") полностью стирает текущую команду, включая и напечатанный текст, если редактор находится в текстовом режиме;

г) нажатие 2 раза комбинации клавиш СКЛ и Р запускает редактор сначала.

После удаления неправильных данных пользователь может продолжить работу.

4.12. Остановы

Если возникает непредусмотренное прерывание, то происходит останов (см.табл.16). По адресу останова можно определить причину останова.

Таблица 16

Адрес останова	Причина останова
12	Прерывание по резервной команде
16	Прерывание по Т-разряду
26	Прерывание по нарушению питания
32	Прерывание по команде EMT
36	Прерывание по команде TRAP
40	Ошибка, обнаруженная программой "Супервизор ввода-вывода"

5. ПРОГРАММА ОТЛАДКИ

5.1. Назначение

Программа отладки (ПО) предназначена для отладки программ пользователя в формате загрузочных модулей путем прогонки их определенными участками и проверки на ожидаемые результаты в различных точках.

Отладка программ осуществляется в режиме диалога пользователя с ЭВМ. Во время отладки необходимо иметь листинг отлаживаемой программы.

Небольшие изменения можно вносить в программу в процессе отладки. Затем программа может быть выполнена под управлением ПО для проверки правильности произведенных изменений. Значительные изменения, такие как введение пропущенной подпрограммы, должны быть внесены в исходную программу перед новой трансляцией.

5.2. Определения

Открытой ячейкой является ячейка, содержимое которой-слово или байт- напечатано по команде ПО для просмотра и может быть изменено.

Закрытой ячейкой является ячейка, содержимое которой недоступно для изменения.

Точкой останова является адрес, по которому ПО прекращает выполнение программы пользователя.

5.3. Команды и функции программы отладки

Входной информацией для программы отладки являются команды пользователя, вводимые с клавиатуры ЭВМ. Пользователь может подать очередную команду после того, как ПО напечатала "." или содержимое открытой ячейки, что сигнализирует об ожидании очередной команды.

После выполнения любой команды (кроме открытия ячеек и регистров) ПО печатает "." .

Команды ПО по их назначению можно разделить на следующие группы:

а) команды открытия и изменения содержимого ячеек;
б) команды управления;
в) команды поиска;
г) команда вычисления смещения;
д) команды задания режима работы.

Полный перечень команд ПО приведен в табл. I7

Таблица I7

Формат команды	Наименование команды
n/	Открыть слово в ячейке n
/	Открыть слово, которое было открыто последним
n\	Открыть байт в ячейке n
\	Открыть байт, который был открыт последним
↓	Открыть следующую ячейку
↑	Открыть предыдущую ячейку
↓↑	Закрыть ячейку
@	Интерпретировать содержимое открытой ячейки как абсолютный адрес и открыть ячейку по этому адресу
_	Интерпретировать содержимое открытой ячейки как индексное слово относительного метода адресации и открыть ячейку по исполнительному адресу
>	Интерпретировать младший байт открытой ячейки как смещение команды ветвления и открыть ячейку, определяемую этим смещением
<	Открыть ячейку, следующую за ячейкой, к которой была подана команда ">", "@" или " _ "
%n/	Открыть регистр общего назначения n (n =0-7)
%S/	Открыть регистр состояния программы
%P/	Открыть ячейку, содержащую приоритет ПО
n;B	Задать точку останова n
n;mB	Задать точку останова n с номером m
;B	Удалить все точки останова
mB	Удалить точку останова с номером m
%B/	Определить точку останова с номером 0
n;G	Запустить программу с адреса n
;P	Продолжить выполнение программы с точки останова

Формат команды	Наименование команды
K;P	Продолжить выполнение программы с точки останова K раз
¤M/	Открыть маску поиска
n;W	Поиск слова n
n;E	Поиск адреса n
;nS	Установить командный режим
;S	Выйти из командного режима
n/¤m;0	Вычислить смещение от ячейки n до ячейки m

Допустимыми вводными знаками являются вышеприведенные команды и восьмеричные цифры. Все буквы, используемые в обозначении команд, набираются на нижнем регистре клавиатуры ЭПМ.

В случае неопознанной или недопустимой команды ПО игнорирует эту команду, печатает

?

.

и ожидает следующую команду.

При печати пользователем чисел первые нули можно опускать. Например, вместо 001000 пользователь может напечатать 1000. Если пользователь напечатал более шести цифр, то ПО воспримет только последние шесть цифр.

ПО не проверяет правильность задания адреса в команде. Например, по команде "177774/" или " ¤20" осуществляется обращение к несуществующей ячейке памяти, что вызывает прерывание по вектору с адресом 000004.

5.4. Команды открытия и изменения содержимого ячеек

С помощью команд ПО можно распечатать и, если необходимо, изменить содержимое регистра или любой ячейки памяти. Для этого используются команды " n/ ", "/", " n\ ", "\", " ↑ ", " ↓ ", " → ",

" @ ", " _ ", " > ", " < ", " ¤n ", " ¤S ", " ¤p ".

5.4.1. Открыть слово в ячейке n

Формат команды: n/

По команде " n/ " открывается ячейка с адресом n.

Пример:

. 1000/012746

В данном примере в ответ на точку пользователем была подана команда "1000/". Ячейка 1000 открыта для рассмотрения и изменения. Её содержимое - 012746.

5.4.2. Открыть байт в ячейке n

Формат команды: n\

Кроме слов ПО оперирует также с байтами.

Пример:

. 1001\025 Открыт байт в ячейке 1000

Если перед подачей команды " \ " было открыто слово, то по команде " \ " будет открыт младший байт этого слова.

Пример:

. 1002/012740\340 Открыт младший байт

5.4.3. Закрыть ячейку

Формат команды: ↵

Если содержимое ячейки изменять не надо, пользователь подает команду " ↵ ". Открытая ячейка закрывается, ПО печатает точку и ждет следующую команду.

Если содержимое ячейки необходимо изменить, пользователь печатает новое содержимое и подает команду " ↵ ".

Пример:

. 1000/012746 ↵ Содержимое ячейки 1000 не изменяется, ячейка закрыта

. 1006/000770 750↵ Содержимое ячейки 1006 изменено на 750, ячейка закрыта

5.4.4. Открыть слово, которое было открыто последним

Формат команды: /

Эта команда предназначена для открытия ячейки, которая была открыта последней. Она дает возможность проверить правильность

изменения содержимого.

Пример:

. 1006/000750 6 ↓

. /000006 Вновь открыта ячейка 1006

5.4.5. Открыть байт, который был открыт последним

Формат команды: \

Эта команда аналогична команде "/", только вместо слова открывается байт.

Примечание. Подача любой команды ПО в то время, когда ячейка открыта, приводит к закрытию этой ячейки.

5.4.6. Открыть следующую ячейку

Формат команды: ↓

Пример:

. 1000/012746 ↓ Ячейка 1000 закрыта, ее содержимое не изменено. Ячейка 1002 открыта

001002/012740

Команда " ↓ " оперирует также с байтами.

Пример:

. 1001\025 ↓ Открыть следующий байт

001002\340

5.4.7. Открыть предыдущую ячейку

Формат команды: ←

Пример:

. 1002/012740 ←

001000/012746 Открыта ячейка 1000

Команда " ← " оперирует также с байтами.

Пример:

. 1002 \340 ← Открыть предыдущий байт

001001 \025

Изменение открытой ячейки можно производить непосредственно перед подачей команд " ↓ ", " ← ", " _ ".

Пример:

. 100/000222 4 ↓

000102/000111 6 ←

000100/000004 100 _

000202/177776

5.4.8. Интерпретировать содержимое открытой ячейки как абсолютный адрес и открыть ячейку по этому адресу

Формат команды: @

Пример:

. 1050/001024 @	Абсолютный адрес 1024
001024/000500 ≠	Ячейка 1024 открыта и закрыта
. 1006/000006 2100 @	Изменено содержимое на 2100
002100/177774	Открыта ячейка 2100

5.4.9. Интерпретировать содержимое открытой ячейки как индексное слово относительного метода адресации и открыть ячейку по исполнительному адресу

Формат команды: _

Пример:

. 1006/000006 _	Индексное слово – 6
001016/100405	Открыта ячейка 1016

5.4.10. Интерпретировать младший байт открытой ячейки как смещение команды ветвления и открыть ячейку, определяемую этим смещением

Формат команды: >

Пример:

. 1032/000407 301 >	Смещение равно – 77
000636/000010 ≠	Следующая открываемая ячейка 000636
.	(1034-(77x2))

5.4.11. Открыть ячейку, следующую за ячейкой, к которой была подана команда ">", "@" или "_"

Формат команды: <

Если ни одна из команд ">", "@" или "_" не подавалась до подачи команды "<", то действие команды "<" аналогично действию команды "≠".

Пример:

. 1032/000407 301 >	
000636/000100 <	По команде "<" открывается
001034/002040 @	ячейка 1034, следующая за ячейкой 1032

002040/000405 < По команде " < " открывается
001036/020102 < ячейка 1036
001040/000001

Команда " < " оперирует также с байтами.

Пример:

. 1034/002040 @
002040/000405\005 < Вновь открыт байт
001035\004 < Команда " < " оперирует с байтами
001036\102

5.4.I2. Открыть регистр общего назначения n

Формат команды: ¤n/

где n=0,1,...,7

Пример:

. ¤0/000033 ↓ Открыт R0 и закрыт
. ¤4/000474 464 ↓ Содержимое R4 изменено, регистр закрыт

Правильность изменения содержимого регистра R4 можно проверить по команде "/".

Пример:

. ¤4/000474 464 ↓
. /000464

С содержимым регистров общего назначения могут оперировать команды " ↓ ", " ¬ ", " – ", " @ ".

5.4.I3. Открыть регистр состояния программы

Формат команды: ¤S/

По команде " ¤S/ " ПО печатает содержимое регистра состояния программы (РСП) пользователя.

Пример:

. ¤S/000311 ↓ Открыт и закрыт РСП

5.4.I4. Открыть ячейку, содержащую приоритет ПО

Формат команды: ¤P/

По команде " ¤P/" открывается ячейка, содержащая приоритет ПО. Она может содержать величину от 0 до 7. Первоначальное содер-

жимое этой ячейки – 7.

Если в ячейку, содержащую приоритет ПО, записана величина 377, то после выполнения команды "n;G" или "K;P" (см.п.5.5) приоритет ПО установится равным приоритету программы пользователя

Пример:

```
.  @P/000006     377^
.
```

5.5. Команды управления

Программа пользователя выполняется под управлением ПО. Это достигается командами задания точек останова "n;B", "n;mB", " ;mB", " @B/" и запуска программы "n;G", " ;P", "K;P".

С помощью ПО можно задать в программе пользователя 8 точек останова с номерами 0–7. В качестве точек останова нельзя использовать ячейки, в которых находятся данные или команды, вызывающие прерывание программы IOT, EMT, TRAP и т.д. Ячейки, содержащие команды прерывания, могут использоваться в качестве точек останова лишь в том случае, если выходы из подпрограмм обработки этих прерываний осуществляются по команде RTI. В противном случае управление ПО не будет передано.

5.5.1. Задать точку останова n с номером m

Формат команды: n;mB

По команде "n;mB" задается точка останова n с номером m (m=0, 1, ...7).

Если точки останова задаются последовательно, то номера точек останова можно не задавать, т.е. для задания точек останова использовать команду " n;B".

Пример:

```
.  1020;B     Задана  точка останова  1020  с номером  0
.  1030;B     Задана  точка останова  1030  с номером  1
.  1040;B     Задана  точка останова  1040  с номером  2
.  1032;1B    Удалена точка останова 1030  с номером  1
              Задана  точка останова  1032  с номером 1
```

5.5.2. Удалить все точки останова

Формат команды: ;B

По команде " ;B" удаляются все точки останова программы пользователя. По команде " ;mB" удаляется точка останова с номером m (m=0,1, ...7).

Пример:

. ;3B Удалена точка останова с номером 3

.

5.5.3. Определить точку останова с номером 0

Формат команды: ИВ/

По команде " ИВ/" открывается ячейка, содержащая точку останова с номером 0. Точки останова с номерами 1-7 могут быть напечатаны по порядку по команде " ↓ ". Девятая ячейка этой последовательности содержит адрес останова в командном режиме (см.п.5.8.1). Если точка останова не задана, то печатается адрес ячейки П0. Точку останова можно заменить, напечатав новое содержимое.

Пример:

. ИВ/001020 ↓ Точка останова с номером 0

пппппп/001032 ↓ Точка останова с номером 1

пппппп/(адрес ячейки П0) Точка останова с номером 2

.

.

.

В данном примере точка останова с номером 2 не задана. Последующие восемь ячеек распечатанной последовательности содержат число повторений для каждой точки останова. Девятая ячейка в этой последовательности содержит число выполняемых команд в командном режиме. Число повторений задается по команде "K;P" (см.п.5.5.5.) и определяет выполнение программы пользователя заданное число раз до соответствующей точки останова .

.

.

.

пппппп/001036 ↓ Точка останова с номером 7

пппппп/пппппп ↓ Адрес останова в командном режиме

nnnnnn /000000 15‡ Число повторений для точки останова с номером 0

nnnnnn/000000 Число повторений для точки останова с номером 1 не задано

В этом примере nnnnnn – адреса ячеек ПО.

Если для точки останова число повторений равно К, то программа пользователя будет выполняться от этой точки останова до следующей точки останова К раз. Первоначально число повторений для всех точек останова равно 0. Если точка останова задана, а число повторений для этой точки не задано, то ПО устанавливает его равным 1.

5.5.4. Запустить программу с адреса n

Формат команды: n;G

Программа пользователя будет выполняться с адреса, указанного в команде "n;G" до выхода на точку останова или останов. Когда встречается точка останова, выполнение программы пользователя прекращается, ПО получает управление и печатает:

Bm;n

где m – номер точки останова

n – точка останова.

Пример:

. 1020;0B	Задана точка останова с номером 0
. 1000;G	Программа пользователя выполняется с адреса
B0;1020	1000 до точки останова 1020
.	

Если при выполнении программы пользователя встречается неизвестная точка останова, то ПО печатает BE и точку останова. Неправильная установка точки останова вызывается использованием в программе пользователя команды BPT или прерывания по Т-разряду.

Пример:

BE 001002 Неправильно установлена точка останова 1002

.

5.5.5. Продолжить выполнение программы с точки останова

Формат команды: ;P

По команде ";P" ПО передает управление на продолжение выполнения программы пользователя с точки останова. По команде "K;P" программа пользователя выполняется с текущей точки останова до следущей точки останова K-1 раз и останавливается на этой точке останова, встретив ее в K-ий раз; ПО получает управление и печатает Bп;п (см.п.5.5.4).

Пример:

1010;B	Задана точка останова 1010 с номером 0
1070;B	Задана точка останова 1070 с номером 1
1000;G	Программа пользователя выполняется с адреса 1000
B0;001010	Выполнение программы прекращается на точке останова с номером 0
4;P	Программа циклится от точки останова с номером 0 до точки останова с номером 1 9 раза и останавливается на точке останова с номером 1, встретив ее в 4-ый раз
B1;1070	

Команда "K;P" должна подаваться только после выполнения команд "п;B" и "п;G".

5.6. Команды поиска

ПО позволяет производить поиск слов или адресов в заданном массиве программы пользователя. Для этого используются команды "M/", "п;M", "п;E".

5.6.1. Открыть маску поиска

Формат команды: M/

По команде "M/" открывается маска поиска (логический множитель) для поиска слов или адресов. Следующие две ячейки ПО, которые можно открыть по команде "}", содержат нижний и верхний пределы поиска. Задать и изменить маску и пределы поиска можно обычным способом. Первоначально эти ячейки содержат нули.

Пример:

M/000000	177400	} Задана маска
000000/000000	1000	} Задан нижний предел

nnnnnn /000000 1040 Задан верхний предел

Здесь nnnnnn - адреса ячеек ПО.

5.6.2. Поиск слова n

Формат команды: n;W

Эта команда предназначена для поиска слова n в заданном массиве. Перед подачей команды " n ;W " должны быть определены маска и пределы поиска (см.п.5.6.I.). Во время поиска просматриваются те разряды, которые в маске установлены в "1". Над каждым выбранным словом и словом поиска n производится операция "Исключающее ИЛИ", а над результатом этой операции и маской поиска производится операция "Логическое И". Если результат операции равен нулю, то соответсвующее слово найдено, и ПО печатает адрес этого слова и само слово.

Пример:

. ⌘M/000000	177400 ⇟	Задана маска
nnnnnn/000000	1000 ⇟	Задан нижний предел
nnnnnn/000000	1040 ⇟	Задан верхний предел
. 400;W		Слово поиска 400 находится в
001010/000770		заданном массиве в ячейках
001034/000404		1010 и 1034
.		

Если маска поиска равна нулю, то распечатываются все ячейки заданного массива.

Пример:

. ⌘M/177400	0 ⇟	Маска поиска равна нулю
nnnnnn/001000	⇟	Нижний предел 1000
nnnnnn/ 001040	1006 ⇟	Верхний предел 1006
. 400;W		
001000/010706		Распечатываются все ячейки задан-
001002/005746		ного массива
001004/012700		
001006/177560		

5.6.3. Поиск адреса n

Формат команды: n;E

Эта команда предназначена для поиска адреса n в заданном массиве. Перед подачей команды " n;E " должны быть определены маска и пределы поиска (см:п.5.6.I.). Ячейки, содержащие абсолютный

адрес, индексное слово относительно метода адресации или команду ветвления к заданному адресу, будут распечатаны.

Пример:

```
. ЖМ/177400 ↓
nnnnnn/001000  1010 ↓
nnnnnn/001040  1060 ↓
. 1034;E                    Поиск адреса 1034
001016/001006               Команды ветвления к адресу 1034
001054/002767               находятся в ячейках 1016 и 1054
. 1020;E                    Поиск адреса 1020
001022/177774               Индексное слово в ячейке 1022
001030/001020               Абсолютный адрес в ячейке 1030
.
```

5.7. Команды вычисления смещения

Относительная адресация и переходы требуют применения смещения-числа слов или байт вперед или назад от адреса текущей ячейки в сторону исполнительного адреса. Для этого используется команда "*n*/X*m*;0".

5.7.1. Вычислить смещение от ячейки *n* до ячейки *m*

Формат команды: *n*/X*m*;0 ,

где X- содержимое ячейки *n*.

По команде "*n*/X*m*;0" печатается смещение в байтах и словах от ячейки *n* до ячейки *m*. Смещение в словах печатается лишь в том случае, если смещение в байтах четное. В процессе отладки содержимое ячейки *n* можно изменять.

Пример:

```
. 346/000034  414;0          000044    000022      22 ↓
. /000022
```

В этом примере ячейка 346 открыта и печатается смещение от этой ячейки до ячейки 414, где 44- смещение в байтах, а 22-смещение в словах. Пользователь напечатал новое содержимое ячейки 346 - 22 и проверил правильность записи.

Смещение в команде ветвления можно изменить следующим

образом:

Пример:

```
. 1024/103406     1052;0     000024 012\006  12 ↵
. /103412
```

В данном случае изменено смещение в команде условного ветвления BCS (BLO). ПО напечатала смещение от ячейки 1024 до ячейки 1052. По команде "\" вновь открыт байт ячейки 1024. Пользователь изменяет содержимое ячейки 1024, печатая 12.

5.8. Команды задания режима работы

С помощью команд ПО можно изменить режим работы программы отладки. Для этого используются команды ";nS" и "S".

5.8.1. Установить командный режим

Формат команды: ;nS ,

где n – любое положительное число, используемое для отличия этой команды от команды "S".

По команде ";nS" устанавливается командный режим. Этот режим позволяет выполнить заданное число команд программы пользователя, независимо от установленных точек останова, т.е. при выполнении программы пользователя точки останова будут игнорироваться. Программа пользователя начинает выполняться после установки режима и подачи команды "n;G" (см.п.5.5.4.). Выполняется одна команда программы пользователя, если не установлена ячейка, содержащая число выполняемых команд в командном режиме (см.п.5.5.3.), управление передается ПО и печатается

B8;n ,

где n – адрес следующей команды, которая должна быть выполнена. Этот адрес записывается в ячейку, следующую за ячейкой, содержащей точку останова с номером 7.

Выполнение программы пользователя можно продолжить, подав команду "K;P", где K определяет число команд, которое необходимо выполнить.

5.8.2. Выйти из командного режима

Формат команды: ;S

По команде ";S" ПО выходит из командного режима.

5.9. Загрузка и запуск

Программа отладки находится на перфоленте в виде объектного

модуля.

Загрузочный модуль ПО можно получить совместно с программой пользователя или отдельно.

В первом случае объектный модуль программы отладки и объектные модули программы пользователя компонуются с помощью программы "Компоновщик" (см. раздел 3) в единый загрузочный модуль. Полученный загрузочный модуль вводится в память ЭВМ с помощью Абсолютного загрузчика (см. п.6.2) и автоматически запускается с адреса загрузки ПО (метка O.ODT), печатая на ЭПМ точку.

Во втором случае загрузочный модуль программы отладки получается отдельно и вводится в память ЭВМ с помощью Абсолютного загрузчика после загрузки программы пользователя и автоматически запускается с адреса загрузки (метка O.ODT), печатая на ЭПМ точку.

5.IO. Повторный запуск

Повторный запуск ПО выполняется с адреса загрузки +2 или адреса загрузки +4.

При запуске ПО с адреса загрузки +2 сохраняется содержимое РОН программы пользователя и удаляются все точки останова, заданные пользователем , т.е. выполняется команда ",B" (см. п.5.5.2.) и печатается точка.

При запуске ПО с адреса загрузки +4 сохраняется содержимое РОН программы пользователя, запоминаются и удаляются все точки останова, заданные пользователем, которые могут быть в дальнейшем восстановлены по команде "*n;G* " (см. п.5.5.4.), и печатается "BE".

6. ПРОГРАММЫ ЗАГРУЗКИ

6.I. Программа начальной загрузки

6.I.I. Назначение

Программа начальной загрузки выполняет загрузку двоичных программ, отперфорированных в начальном формате и содержащих не более 162_8 слов. Одной из таких программ является программа "Абсолютный загрузчик", с помощью которой загружается большинство программ.

6.I.2. Распределение памяти

Программа начальной загрузки вводится в память ЭВМ с помощью переключателей клавишного регистра (КР) пульта управления ЭВМ. Программа обычно располагается в памяти с адреса ХХ7744 по ХХ7776, где ХХ определяется емкостью используемой памяти (см.табл.I8).

Таблица I8

ХХ	Начальный адрес	Память (К байт)
0I	0I7744	8
03	037744	I6
05	057744	24
07	077744	32
II	II7744	40
I3	I37744	48
I5	I57744	56

6.I.3. Текст программы

Ниже приводится текст программы начальной загрузки на языке Ассемблера для памяти емкости 16К байт.

```
                        ;  программа начальной загрузки
        000001              R1 = % 1
        000002              R2 = % 2
        037400              LOAD = 37400
        037744                 . = 37744
037744  016701  START:     MOV DEVICE, R1   ; адрес регистра
037746  000026                              ; устройства ввода
037750  012702  LOOP:      MOV #.-LOAD+2,R2 ; приращение
037752  000352                              ; адреса в R2
037754  005211  ENABLE:    INC @R1          ; считывание
037756  105711  WAIT:      TSTB @R1         ; ожидание
037760  100376              BPL  WAIT        ; готовности
037762  116162              MOVB 2(R1),LOAD(R2); пересылка
037764  000002                              ; байта в память
037766  037400
037770  005267              INC  LOOP+2      ; увеличение
037772  177756                               ; адреса загрузки на 1
037774  000765  BRNCH:      BR   LOOP        ; повторить цикл
037776  000000  DEVICE:    0                ; адрес регистра
        000001              .END             ; устройства ввода
```

В ячейку DEVICE записывается адрес регистра состояния перфоленточного устройства ввода, используемого при загрузке перфолент:

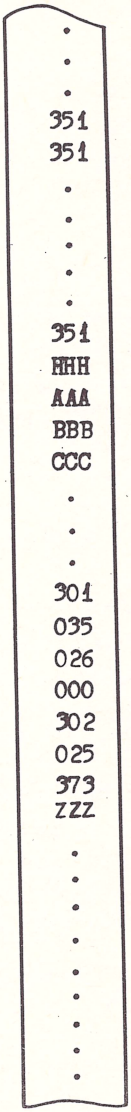
177550 – высокоскоростное перфоленточное устройство ввода
177560 – низкоскоростное перфоленточное устройство ввода

Программа начальной загрузки загружает ленты в любое место памяти между XX7400 и XX7744 и не более 162_8 слов

6.I.4. Формат начальной ленты

Ленту, отперфорированную в начальном формате (см. рис. 1), будем называть **начальной лентой**. Эта лента, в отличие от абсолютных лент (см. п.6.2.4.), начинается пробивкой кода 351, а не

синхродорожки:



специальный код
(≈60 см)

младший байт адреса загрузки без 1
загружаемая программа
(162_8 слова или 344_8 байта)

команды программы
начальной загрузки

смещение перехода

Рис. I. Начальный формат

Младший байт адреса загрузки определяется по формуле :
Адрес загрузки – XX7400

Смещение перехода определяется по формуле:
(Адрес ячейки, которой передается управление - ХХ7776) /2

После загрузки начальной ленты программа начальной загрузки автоматически передает управление ячейке, определяемой смещением перехода.

6.I.5. Ввод программы начальной загрузки

Программа начальной загрузки вводится в память с пульта управления ЭВМ вручную с помощью переключателей клавишного регистра следующим образом:

а) установите на КР код ХХ7744 (ХХ определяется по табл.I8) и нажмите клавишу "ЗАН АДР". В регистре "АДРЕС" индицируется ХХ7744;

б) установите на КР код первой команды 016701 и нажмите клавишу "ЗАП". В регистре "ДАННЫЕ" индицируется 016701.

Примечание. Во время ввода данных в последовательные ячейки памяти при нажатии на клавишу "ЗАП" содержимое регистра "АДРЕС" автоматически наращивается на 2 и указывает на адрес ячейки, в которую введены данные.

в) установите на КР код следующей команды 000026 и нажмите клавишу "ЗАП". В регистре "ДАННЫЕ" индицируется 000026;

г) установите на КР код следующей команды, нажмите клавишу "ЗАП" и продолжайте вводить последующие коды команд (убедитесь, что содержимое ячейки ХХ7766 соответствует емкости выбранной памяти) до тех пор, пока в ячейку ХХ7774 не будет введен код 000765;

д) введите в последнюю ячейку ХХ7776 адрес регистра состояния используемого перфоленточного устройства ввода:

177550 - высокоскоростное перфоленточное устройство ввода

177560 - низкоскоростное перфоленточное устройство ввода

Проверка правильности ввода программы начальной загрузки выполняется следующим образом:

е) установите на КР код XX7744 и нажмите последовательно клавиши "ЗАН АДР" и "ЧТЕН". В регистре "ДАННЫЕ" индицируется содержимое ячейки XX7744. Сравните его с правильным содержимым - 016701. В случае несовпадения перейдите к п.6.I.5.и, в случае совпадения выполните следующее:

ж) нажмите клавишу "ЧТЕН". Содержимое ячейки, адрес которой находится в регистре "АДРЕС", индицируется в регистре "ДАННЫЕ". Сравните содержимое регистра "ДАННЫЕ" с правильным содержимым. В случае совпадения повторяйте этот пункт до тех пор, пока не будут проверены все ячейки и перейдите к п.6.I.5.и, если индицируется неправильное содержимое.

Неправильно введенные коды команд можно исправить, выполнив следующее:

з) установите на КР адрес требуемой ячейки и нажмите клавишу "ЗАН АДР". В регистре "АДРЕС" индицируется адрес требуемой ячейки;

и) установите на КР правильный код команды и нажмите клавишу "ЗАП".

Нажав затем клавишу "ЧТЕН", убедитесь, что код команды введен правильно. Он будет индицироваться в регистре "ДАННЫЕ".

6.I.6. Загрузка начальных лент

Начальные ленты загружаются в память ЭВМ следующим образом:

а) установите на пульте управления ЭВМ клавишу "СТАРТ/СТОП" в положение СТОП;

б) заправьте ленту в выбранное перфоленточное устройство ввода, установив её специальным кодом (351) на считывающий элемент и включите его питание;

в) установите на КР код XX7744 - пусковой адрес программы начальной загрузки и нажмите клавишу "ЗАН АДР";

г) установите клавишу "СТАРТ/СТОП" в положение СТАРТ и нажмите клавишу "ПУСК". Лента движется и загружается в память;

д) останов. Регистр "АДРЕС" = адрес загрузки +2. Лента загружена правильно.

Если лента не движется после нажатия на клавишу "ПУСК", то возможными причинами могут быть:

е) неправильно введена программа начальной загрузки;
ж) неправильно выбрано устройство ввода;
з) код 351 находится не на считывающем элементе устройства ввода;
и) загружаемая лента неправильно установлена в устройстве ввода.

6.2. Абсолютный загрузчик

6.2.1. Назначение

Абсолютный загрузчик - это программа, позволяющая загружать в любое место памяти двоичные данные, отперфорированные на ленте в абсолютном формате. Он используется для загрузки программ перфоленточной операционной системы и программ, полученных с помощью Ассемблера и Компоновщика.

Основными функциями Абсолютного загрузчика являются:

а) загрузка программ согласно выбранного режима;
б) проверка правильности загрузки с помощью контрольного суммирования;
в) автоматический запуск загруженной программы по завершении загрузки без дополнительных действий оператора.

6.2.2. Распределение памяти

Программа "Абсолютный загрузчик", отперфорированная в начальном формате, загружается в память с помощью программы начальной загрузки и использует ячейки с XX7474 по XX7722. Значение XX определяется по табл. I8.

6.2.3. Режимы загрузки

Абсолютный загрузчик имеет следующие режимы загрузки:

а) нормальный;
б) перемещаемый;
в) перемещаемый с продолжением.

Все перечисленные режимы загрузки задаются установкой разрядов переключателя режима. На ЭВМ с клавишным регистром переключателем режима является КР, на ЭВМ без клавишного регистра - ячейка XX7516.

Нормальный – это режим загрузки программ в память ЭВМ с адреса, указанного на перфоленте. Этот режим определяется установкой 0-го разряда переключателя режима в "0" перед началом загрузки.

Перемещаемый – это режим загрузки программ в выбранную область памяти ЭВМ. Этот режим используется для загрузки независимых от местоположения в памяти программ. Он определяется установкой переключателя режима:
разряд 00=1,
разряды 15-01 = приращение адреса загрузки.

Перемещаемый с продолжением – это режим загрузки программ с адреса, следующего за конечным адресом последней загруженной программы. Этот режим используется, например, когда загружаемая программа содержится на нескольких лентах (для загрузки в последовательные ячейки памяти необходимо, чтобы адрес загрузки последующих лент был равен нулю). Этот режим определяется установкой переключателя режима:
разряд 00=1
разряды 15-01=00000

6.2.4. Формат абсолютной ленты

Ленту, отперфорированную в абсолютном формате, будем называть абсолютной лентой. Абсолютная лента содержит несколько блоков данных, отперфорированных в абсолютном формате (см. рис. 2). Последним блоком абсолютной ленты является специальный блок из 6 байт.

Количество байтов – это целое число байтов в блоке, начиная с байта "начало блока" до байта "ДКС". Количество байтов вычисляется и перфорируется на ленте с помощью программы "Перфорация в абсолютном формате" или при трансляции программы с помощью ассемблера.

Данные любого блока, имеющего количество байтов (XXXXXX) более 6, загружаются последовательно в память согласно выбранного режима.

Если количество байтов блока равно 6, то адрес загрузки (УУУУУУ) проверяется на четность. Если адрес четный, то Абсолют-

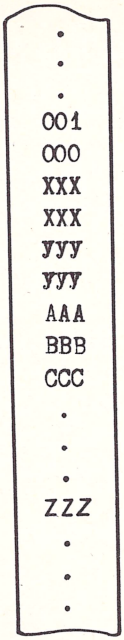


Рис. 2. Блок данных

ный загрузчик будет автоматически передавать управление ячейке с этим адресом. Если адрес загрузки нечетный, то происходит останов по адресу XX7710. Останов по этому адресу означает конец загрузки перфоленты; в режиме перемещения с продолжением останов используется для заправки перфолент. Например, на ленте отперфорировано два блока. Первый блок содержит программу, а второй - 6 байт с четным адресом загрузки. После загрузки перфоленты программа автоматически начнет выполняться с адреса, указанного во втором блоке.

ДКС представляет собой дополнение до 2 младшего байта контрольной суммы (суммы всех предшествующих ДКС байтов блока).

6.2.5. Загрузка Абсолютного загрузчика на ЭВМ с клавишным регистром.

Программа "Абсолютный загрузчик" загружается в память ЭВМ с помощью программы начальной загрузки (см.п.6.I.6).

Примечание. Абсолютный загрузчик всегда загружается в тот же блок памяти, где расположена программа начальной загрузки.

6.2.6. Загрузка абсолютных лент на ЭВМ с клавишным регистром

а) установите в ячейке ХХ7776 (ХХ определяется по таблице I8) адрес регистра состояния используемого перфоленточного устройства ввода.

177550 - высокоскоростное перфоленточное устройство ввода
177560 - низкоскоростное перфоленточное устройство ввода

б) заправьте абсолютную ленту в устройство ввода синхродорожкой на считывающий элемент и включите его питание;

в) занесите в регистр "АДРЕС" код ХХ7500 - пусковой адрес программы "Абсолютный загрузчик";

г) задайте на КР режим загрузки согласно табл.I9;

Таблица I9

КР 15-01	КР00	Режим загрузки
-	0	Нормальный
Приращение адреса загрузки	1	Перемещаемый
00000	1	Перемещаемый с продолжением

Примечание. Приращение адреса загрузки должно быть четным.

д) установите клавишу "СТАРТ/СТОП" в положение СТАРТ;

е) нажмите клавишу "ПУСК". Лента движется и загружается в память;

По окончании загрузки происходит:

ж) управление автоматически передается на выполнение загруженной программы;

з) останов. Регистр "АДРЕС" = XX7712. Младший байт регистра "ДАННЫЕ" равен 000. Лента загружена правильно. В режиме перемещения с продолжением установите в устройство ввода следующую ленту и нажмите клавишу "ПРОД";

и) останов. Регистр "АДРЕС" =XX7612. Младший байт регистра "ДАННЫЕ" не равен 000. Лента загружена неправильно. Повторите загрузку, перейдя к п.6.2.6.б.

6.2.7. Загрузка Абсолютного загрузчика на ЭВМ без клавишного регистра

а) Включите питание ЭВМ.

На ЭПМ печатается содержимое ячейки 000024 (обычно все нули) и "*" ;

б) установите на панели управления ЭВМ клавишу "ПРОГРАММА/ПУЛЬТ" в положение ПРОГРАММА;

в) заправьте перфоленту "Абсолютный загрузчик" в перфоленточное устройство ввода, установив пробивкой кода 351 на считывающий элемент, и включите его питание;

г) напечатайте адрес регистра состояния используемого устройства ввода и " L "

* 177550 L – для высокоскоростного перфоленточного устройства ввода

* 177560 L – для низкоскоростного перфоленточного устройства ввода

д) лента движется и загружается в память. По окончании загрузки на ЭПМ печатается:

* XX7500

*

где XX7500 – пусковой адрес программы "Абсолютный загрузчик" (значение XX определяется по табл.18).

Примечание. Если по окончании загрузки Абсолютного загрузчика клавиша "ПРОГРАММА/ПУЛЬТ" находится в положении ПУЛЬТ, то на ЭПМ печатается

* XX7724

*

где XX7724 – конечный адрес +2 Абсолютного загрузчика.

Для печати пускового адреса Абсолютного загрузчика установите клавишу "ПРОГРАММА/ПУЛЬТ" в положение ПРОГРАММА и напечатайте на нижнем регистре

* P

6.2.8. Загрузка абсолютных лент на ЭВМ без клавишного регистра

Убедитесь, что клавиша "ПРОГРАММА/ПУЛЬТ" находится в положении ПРОГРАММА.

Заправьте абсолютную ленту в выбранное устройство ввода (см.п.6.2.7.) синхродорожкой на считывающий элемент и включите его питание.

Напечатайте на ЭПМ:

а) для загрузки в нормальном режиме

* XX7500G

или

* P

б) для загрузки в перемещаемом режиме

* XX7516/УУУУУУ ZZZZZZ

* XX7500G

или

* XX7516/УУУУУУ ZZZZZZ

* P

где XX7516 – переключатель режима загрузки

ZZZZZZ – приращение адреса загрузки +1 ;

в) для загрузки в перемещаемом режиме с продолжением

* XX7516/УУУУУУ 000001

* P

Лента движется и загружается в память. По окончании загрузки происходит:

г) управление автоматически передается на выполнение загруженной программы;

д) останов. На ЭПМ печатается

* XX7712.

Лента загружена правильно.

В режиме перемещения с продолжением установите в устройство ввода следующую ленту и напечатайте

* P

Для запуска загруженной программы на ЭПМ напечатайте

* XXXXXXG ,

где XXXXXX - пусковой адрес программы

е) останов. На ЭПМ печатается

* XX7612.

Лента загружена неправильно.

Повторите загрузку, перейдя к п.6.2.8.

7. ПРОГРАММЫ ВЫВОДА

7.I. Перфорация в абсолютном формате

7.I.I. Назначение

"Перфорация в абсолютном формате" - это программа, которая выводит на перфоратор содержимое выбранных участков памяти в абсолютном формате.

7.I.2. Распределение памяти

Размещение в памяти программы "Перфорация в абсолютном формате" совместно с загрузчиками приведено на рис.3.

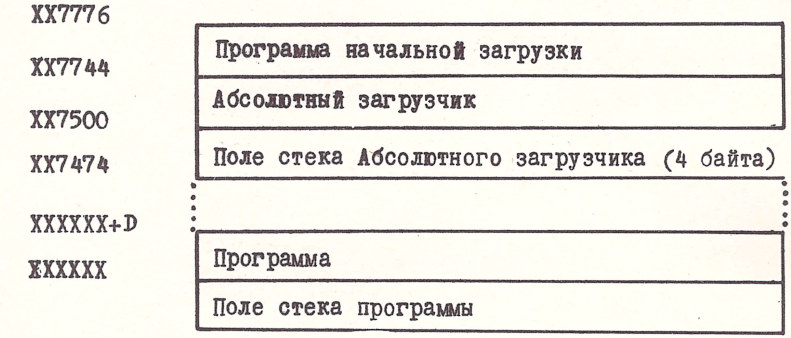


Рис.3. Распределение памяти для программы "Перфорация в абсолютном формате"

На рис. 3:

XX - определяется емкостью используемой памяти (см.табл.I8)

XXXXXX - адрес загрузки программы

D - длина программы (для ЭВМ с клавишным регистром - 204 байта, для ЭВМ без клавишного регистра - 216 байт)

7.I.3. Формат абсолютной ленты

Программа "Перфорация в абсолютном формате" выводит на перфоратор данные поблочно в абсолютном формате (см.п.6.2.). Последним блоком должен быть специальный блок из 6 байт.

7.I.4. Загрузка

Программа "Перфорация в абсолютном формате" загружается в память ЭВМ с помощью Абсолютного загрузчика в перемещаемом режиме с приращением адреса загрузки, равным адресу загрузки (см. п.6.2.).

7.I.5. Работа на ЭВМ с клавишным регистром

а) занесите в регистр "АДРЕС" пусковой адрес программы ХХХХХХ, равный адресу загрузки ;

б) установите клавишу "СТАРТ/СТОП" в положение СТАРТ. Нажмите клавишу "ПУСК";

в) останов. Регистр "АДРЕС" =ХХХХХХ+6.
Установите на КР адрес регистра состояния используемого перфоленточного устройства вывода:
177554 - высокоскоростной перфоратор
177564 - низкоскоростной перфоратор
Нажмите клавишу "ПРОД";

г) останов. Регистр "АДРЕС" =ХХХХХХ+30.
Установите на КР начальный адрес блока выводимого массива. Нажмите клавишу "ПРОД";

д) останов. Регистр "АДРЕС" =ХХХХХХ+36.
Установите на КР конечный адрес блока выводимого массива (адрес последнего байта блока). Нажмите клавишу "ПРОД".
Перфорируется блок данных выбранного массива;

е) останов. Регистр "АДРЕС" =ХХХХХХ+30.
Отперфорирован блок данных выбранного массива.

Для перфорации следующего блока данных перейдите к п.7.I.5.г.

Для перфорации последнего блока (6 байт) выполните следующее:

ж) установите на КР четный адрес загрузки или код 000001.
Нажмите клавишу "ПРОД";
з) останов. Регистр "АДРЕС" =XXXXXX+36.
Установите на КР код 000000. Нажмите клавишу "ПРОД";
Перфорируется последний блок;
и) останов. Регистр "АДРЕС" =XXXXXX+30.
Отперфорирован последний блок. Отперфорированную ленту удалите из устройства вывода и отмаркируйте.

7.I.6. Работа на ЭВМ без клавишного регистра

По окончании загрузки программа "Перфорация в абсолютном формате" запускается автоматически.
а) останов. На ЭПУ печатается
ж XXXXXX+14
где XXXXXX - адрес загрузки
б) запишите в ячейки:
XXXXXX+16/ начальный адрес выводимого массива
XXXXXX+20/ конечный адрес выводимого массива
При перфорации первого массива с использованием низкоскоростного перфоратора
XXXXXX+22/177554 177564
в) включите питание перфоратора
г) напечатайте на нижнем регистре
ж P
Перфорируется один или несколько блоков данных выбранного массива;
д) останов. На ЭПУ печатается
ж XXXXXX+14
Отперфорирован один или несколько блоков данных выбранного массива.
Для перфорации следующего массива данных перейдите к п.7.I.6.6.
Для перфорации последнего блока (байт) выполните следующее:
е) запишите в ячейки:
XXXXXX+16/ четный адрес загрузки или 000001
XXXXXX+20/000000

к) напечатайте на нижнем регистре

↵ P

Перфорируется последний блок;

з) останов. На ЭПМ печатается

↵XXXXXX+14

Отперфорирован последний блок. Отперфорированную ленту удалите из перфоратора и отмаркируйте.

Примечания: 1. Для повторного запуска программы следует напечатать на ЭПМ

↵ XXXXXX+66

2. Первоначально в ячейке XXXXXX+22 содержится адрес регистра состояния высокоскоростного перфоратора - 177554.

7.2. Восьмеричная печать

7.2.1. Назначение

"Восьмеричная печать" - это программа, которая выводит на печатающее устройство ЭПМ или высокоскоростное печатающее устройство содержимое выбранных участков памяти в восьмеричном коде.

7.2.2. Распределение памяти

Размещение в памяти программы "Восьмеричная печать" совместно с загрузчиками приведено на рис.4.

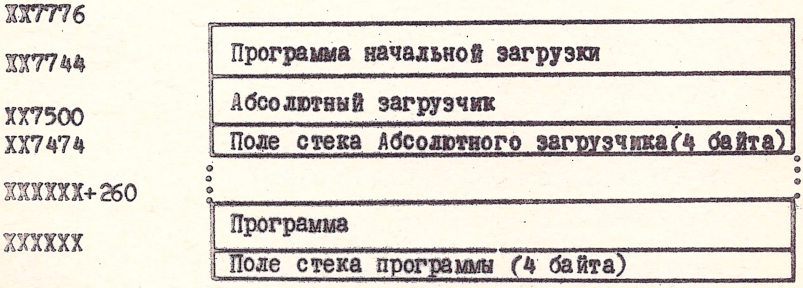


Рис. 4. Распределение памяти для программы "Восьмеричная печать".

На рис.4: XX - определяется емкостью используемой памяти (см.табл.I8);
XXXXXX - адрес загрузки программы

7.2.3. Формат печати

Программа "Восьмеричная печать" выдает на печатающее устройство данные массива, начальный и конечный адреса которого задаются на КР, построчно, т.е. в виде:

аааааа ссссс ..... ссссс
(8 слов)

где аааааа - адрес данных строки
ссссс - данные массива, первое из которых содержится в ячейке аааааа

7.2.4. Загрузка

Программа "Восьмеричная печать" загружается в память ЭВМ с помощью Абсолютного загрузчика в перемещаемом режиме с приращением адреса загрузки, равным адресу загрузки (см.п.6.2.).
Примечание. Перфолента программы "Восьмеричная печать" имеет адрес загрузки равный нулю.

7.2.5. Работа с программой

а) занесите в регистр "АДРЕС" пусковой адрес программы XXXXXX, равный адресу загрузки;
б) установите клавишу "СТАРТ/СТОП" в положение СТАРТ. Нажмите клавишу "ПУСК";
в) останов. Регистр "АДРЕС" =XXXXXX+6.
Установите на КР адрес регистра состояния печатающего устройства
177564 - печатающее устройство ЭПМ
177554 - высокоскоростное печатающее устройство
Нажмите клавишу "ПРОД";
г) останов. Регистр "АДРЕС" =XXXXXX+30.
Установите на КР начальный адрес выводимого массива. Нажмите клавишу "ПРОД";
д) останов. Регистр "АДРЕС" =XXXXXX+36.
Установите на КР конечный адрес выводимого массива. Нажмите клавишу "ПРОД". Данные выбранного массива выводятся на печат

щее устройство;

е) останов. Регистр "АДРЕС" =XXXXXX+30.
Вывод массива закончен.

Для вывода на печать данных последующих массивов перейдите к п.7.2.5.г.

8. СУПЕРВИЗОР ВВОДА-ВЫВОДА

Программа "Супервизор ввода-вывода" предназначена для организации обмена данными между программами пользователя и периферийными устройствами (ЭПМ, низкоскоростными и высокоскоростными перфоленточными устройствами ввода-вывода, высокоскоростным печатающим устройством). Супервизор обеспечивает асинхронное обслуживание периферийных устройств.

Вызов супервизора осуществляется по командам супервизора, формат которых приведен ниже:

```
IOT
.WORD <адрес>
.BYTE <код команды ,  номер позиции>
.WORD <адрес выполнения>
```

Команды супервизора по своему назначению делятся на команды без передачи данных (Восстановление, Начало, Повторный пуск, Поиск, Ожидание) и команды передачи данных (Чтение, Запись, Чтение РВ, Запись РВ). Передача данных осуществляется в различных режимах (форматном: символьном и двоичном, неформатном: символьном и двоичном).

После выполнения команды супервизора управление передается программе пользователя, вызвавшей супервизор.

8.1. Распределение периферийных устройств

Супервизор удовлетворяет заявки программ пользователя на обмен данными с периферийными устройствами. Обращение к периферийным устройствам осуществляется по командам супервизора с помощью номера позиции. Номер позиции - номер строки таблицы I.DAT, содержащей код устройства (см.табл.20). Таблица I.DAT формируется супервизором с помощью команд Восстановление и Начало. Команда Начало единственная команда, использующая код устройства при обращении к периферийному устройству.

Такая организация обращения к периферийным устройствам позволяет делать программу пользователя независимой от устройств.

Таблица 20

Устройство	Код устройства
Клавиатура ЭПМ	1
Печатающее устройство ЭПМ	2
Низкоскоростное перфоленточное устройство ввода	3
Низкоскоростной перфоратор	4
Высокоскоростное перфоленточное устройство ввода	5
Высокоскоростной перфоратор	6
Высокоскоростное печатающее устройство	10

8.I.I. Команда Восстановление

Формат команды

```
IOT
.WORD 0
.BYTE 2,0
```

Команда Восстановление должна предшествовать всем командам супервизора. По этой команде устанавливаются внутренние таблицы супервизора и периферийные устройства в исходное состояние, разрешается прерывание от клавиатуры ЭПМ и присваиваются номера позиций клавиатуре и печатающему устройству ЭПМ "0" и "1" соответственно.

8.I.2. Команда Начало

Формат команды:

```
IOT
.WORD <адрес ячейки, содержащей код устройства>
.BYTE 1,<номер позиции>
```

По команде Начало за устройством закрепляется номер позиции (0-7). Прерывание от устройства запрещается.

Пример:

```
.INIT  = 1
IOT
.WORD  ADRL   ;  присвоение номера позиции "2"
.BYTE  INIT, 2 ;  низкоскоростному перфоратору
```

```
        .
        .
        .
ADRL : .BYTE 4  ;  код низкоскоростного перфоратора
```

8.2. Организация буфера для передачи данных

Для передачи данных необходим буфер. Буфер используется для запоминания данных, хранения сведений о количестве, режиме и состоянии передаваемых данных.

Часть буфера, предшествующая данным, называется заголовком буфера. Адрес буфера (адрес начала заголовка) указывается во втором слове команды передачи данных. Структура буфера приведена на рис. 5

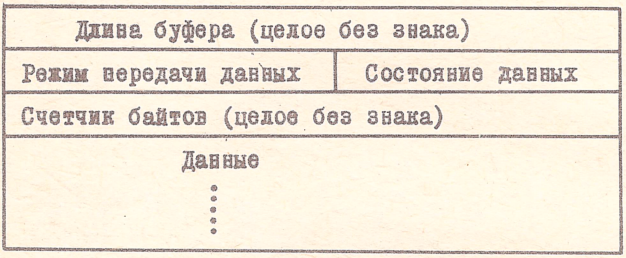


Рис. 5 Структура буфера

Примечание. Программа пользователя не должна изменять заголовок буфера во время передачи данных.

8.2.I. Длина буфера

Первое слово заголовка буфера формируется пользователем. Оно содержит длину буфера в байтах. Супервизор при вводе данных не запомнит байтов больше этого числа. При выводе данных длина буфера не имеет значения.

8.2.2. Байт режима

Младший байт второго слова заголовка буфера (байт режима) формируется пользователем и содержит информацию о режиме передачи данных. В супервизоре предусмотрено четыре режима передачи данных (см.табл.2I). Режимы передачи данных описаны в п.8.3.

Таблица 21

Режим передачи	Значение байта
Форматный символьный	0 (или 200 без "ЭХО")
Форматный двоичный	1
Неформатный символьный	2 (или 202 без "ЭХО")
Неформатный двоичный	3

Операция "ЭХО" применима только к клавиатуре ЭПМ и означает вывод на печать вводимых с клавиатуры данных. Передача данных от одного устройства к другому не включает в себя операцию "ЭХО".

Структура байта режима приведена на рис.6

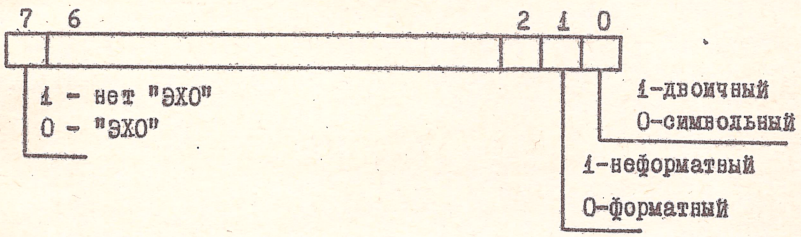


Рис.6. Структура байта режима

8.2.3. Байт состояния

Старший байт второго слова заголовка буфера (байт состояния) формируется супервизором и содержит информацию о состоянии передаваемых данных. Структура байта состояния приведена на рис.7

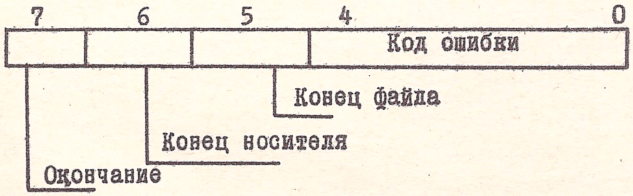


Рис.7. Структура байта состояния

Разряды байта состояния устанавливаются супервизором следующим образом:

а) разряд Окончание

Разряд Окончание устанавливается в "1" при завершении передачи данных:

б) разряд Конец носителя

Разряд Конец носителя устанавливается в "1" при следующих условиях:

для перфоленточных устройств ввода - нет перфоленты, автономный режим, нет питания;

для перфоленточных устройств вывода - нет питания;

для высокоскоростного печатающего устройства - нет бумаги, нет питания;

в) разряд Конец Файла.

Разряд Конец файла устанавливается в "1" при попытке ввода с устройства, для которого разряд Конец носителя установлен в "1". Данные в буфер не вводятся. При выводе разряд Конец файла не устанавливается;

г) коды ошибок

При вводе данных в форматных режимах могут возникнуть ошибки, не вызывающие останова программы. В форматном двоичном режиме возможны ошибки "несовпадение контрольных сумм", "длинная строка", "неправильный режим".

В форматном символьном режиме возможна ошибка "длинная строка". Коды этих ошибок приведены ниже:

2 - несовпадение контрольных сумм;

3 - длинная строка;

4 - неправильный режим.

Ошибка "несовпадение контрольных сумм" возникает при несовпадении контрольной суммы, подсчитанной при вводе, с контрольной суммой блока данных на ленте.

Ошибка "длинная строка" возникает, если двоичный блок или строка знаков больше длины буфера, указанной в заголовке. В этом случае данные, не помещающиеся в буфер, супервизор вводит в последний байт буфера до конца двоичного блока или пока не встретится ограничитель строки (знак ПС или ПФ).

Ошибка "неправильный режим" возникает, если первый ненулевой

код отличен от 001 (признак начала блока), считывание прекращается.

8.2.4. Счетчик байтов

Третье слово заголовка буфера является счетчиком байтов.

Ввод. В неформатных режимах счетчик байтов формируется пользователем и указывает на количество вводимых байтов данных. В форматных режимах счетчик байтов формируется супервизором и указывает количество введенных байтов данных.

Во всех режимах, если установится в "1" разряд Конец носителя, счетчик байтов содержащий число, равное количеству действительно введенных байтов данных.

Если установится в "1" разряд Конец файла, счетчик байтов очищается.

Вывод. Во всех режимах выводится столько байтов данных, сколько указано в счетчике байтов. При выводе супервизор не изменяет содержимое счетчика байтов.

8.3. Режимы передачи данных

8.3.1. Форматный символьный режим

Форматный символьный режим используется для ввода-вывода строки символов (ограничители строки-символы ПС или ПФ).

Если при вводе данных была ошибка "длинная строка", то последний байт данных буфера не содержит кода ограничителя строки.

Функции устройств:

а) клавиатура

Коды символов считываются с клавиатуры ЭПМ и вводятся в буфер. Вводимые символы печатаются на ЭПМ. Некоторые символы выполняют определенные действия (см.табл.22). Операция "ЭХО" может быть запрещена установкой 7-го разряда байта режима.

Таблица 22

Символ	Код	Действие
ПУС	000	Игнорируется
СКЛ/I	011	Печатается 8 пробелов
ЗБ	177	Печатается "\". Стирает последний введенный знак текущей строки. Если для стирания нет никакого символа, то ЗБ игнорируется

Продолжение табл.22

Символ	Код	Действие
СКД/U	025	Печатается " ¬ U ". Стирает текущую строку
ВК	015	Вызывает возврат каретки и перевод строки. Коды символов ВК и ПС вводятся в буфер
СКД/Р	020	Печатается " ¬ Р". Вызывает переход по адресу повторного пуска, указанному в команде Повторный пуск

Символ ПУС и символы стирания ЗБ и СКД/U в буфер не вводятся. Если происходит переполнение буфера, то символы, за исключением ВК и ПС, вводятся в последний байт, но не печатаются. ВК и ПС при переполнении буфера выполняются, но в буфер вводится только ПС. При этом в байт состояния записывается код ошибки "длинная строка". Пример вводимых с клавиатуры символов см. в табл.23 (длина буфера 5 байт)

Таблица 23

Вводимые с клавиатуры знаки	Операция "ЭХО"	Введенные в буфер знаки
ABC ⏎ ABCD ⏎ ABCDEF ⏎ ABCDEF ЗБ ⏎ СКД/U ЗБ ⏎ ABCDEF ЗБ ЗБ ⏎ ABCDEF ЗБ ЗБ ЗБ X ⏎	ABC ⏎⏎ ABCD ⏎⏎ ABCD ⏎⏎ ABCD \ ⏎ ¬ U ⏎⏎ ABCD \\ ⏎⏎ ABCD \\\ ⏎⏎	ABC ⏎⏎ ABCD ⏎ ABCD ⏎ ABC ⏎⏎ ⏎⏎ AB ⏎⏎ AX ⏎⏎

б) низкоскоростное и высокоскоростное перфоленточные устройства ввода

Коды символов вводятся в буфер за исключением символов ПУС и ЗБ, которые игнорируются.

в) печатающее устройство ЭПМ.

Коды символов выводятся из буфера, за исключением символа

ПУС, который игнорируется. Код символа СКЛ/I вызывает печать пробелов.

г) низкоскоростной и высокоскоростной перфораторы

Коды символов выводятся из буфера, за исключением символа ПУС, который игнорируется. За кодом символа СКЛ/I супервизор выводит код символа ЗБ.

д) высокоскоростное печатающее устройство

Коды символов выводятся из буфера, за исключением ПУС и ВК, которые игнорируются. Код символа СКЛ/I вызывает печать пробелов. Символы ПС и ПФ вызывают возврат каретки и перевод строки. Строка символов не должна превышать 80 символов. Все символы, стоящие после первых 80 символов, игнорируются за исключением ПС и ПФ.

8.3.2. Форматный двоичный режим

Форматный двоичный режим используется для ввода-вывода блока данных с перфоленты в абсолютном формате (см.рис.8)

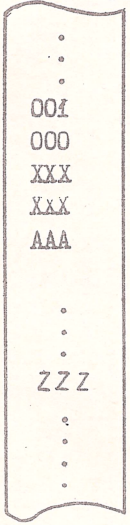


Рис.8. Блок данных

Данные в абсолютном формате вводятся поблочно с контрольным суммированием.

Вывод данных начинается со второго слова заголовка буфера (код 001-байт режима, код 000-байт состояния). Количество байтов

блока супервизор выводит на 4 больше, чем в счетчике байтов заголовка буфера. Последним байтом блока данных выводит ДКС.

8.3.3. Неформатный символьный режим

Неформатный символьный режим используется для ввода-вывода символов по счетчику байтов заголовка буфера.

Функции устройств:

а) клавиатура

Коды символов считываются с клавиатуры и вводятся в буфер. Вводимые символы печатаются на ЭПМ. Некоторые символы выполняют определенные действия (см.табл.24).

Таблица 24

Символ	Код	Действие
СКЛ/I СКЛ/P	011 020	Печатается 8 пробелов Печатается " _ P". Осуществляется повторный пуск программы пользователя с адреса, указанного в команде Повторный пуск

8.3.4. Неформатный двоичный режим

Неформатный двоичный режим используется для передачи восьмиразрядных кодов и кодов символов преобразования (например символов RAD50). Как при вводе так и при выводе счетчик байтов заголовка буфера определяет число передаваемых символов.

8.4. Передача данных

8.4.1. Команда Чтение

Формат команды

```
 IOT
.WORD      <адрес буфера>
.BYTE  11,<номер позиции>
```

По команде Чтение супервизор считывает в режиме прерывания данные с устройства, определенного номером позиции, в соответствии с информацией заголовка буфера.

Начиная передачу данных, супервизор очищает байт состояния, разрешает прерывание от устройства (когда устройство свободно или нет конфликта) и возвращает управление в вызвавшую считывание

программу. По окончании передачи данных супервизор устанавливает соответствующие разряды байта состояния и содержимое счетчика байтов на число байтов, введенных в буфер.

8.4.2. Команда Запись

Формат команды

```
 IOT
.WORD   <адрес буфера>
.BYTE   12, <номер позиции>
```

По команде Запись супервизор выводит в режиме прерывания данные в устройство, определенное номером позиции, в соответствии с информацией заголовка буфера. Количество передаваемых данных определяется счетчиком байтов.

Начиная передачу данных, супервизор разрешает прерывание от устройства (когда устройство свободно или нет конфликта) и возвращает управление в вызвавшую вывод программу. По окончании передачи данных супервизор устанавливает соответствующие разряды байта состояния.

Если при выполнении команды Запись установится разряд Конец носителя, то нельзя определить какое количество данных было выведено.

8.4.3. Взаимодействие устройств при передаче данных

Некоторые периферийные устройства не могут быть использованы в работе одновременно из-за физической связи между ними. Перед выполнением передачи данных супервизор определяет: занято требуемое устройство или находится в работе устройство, которое может помешать работе требуемого устройства, т.е. вызвать конфликт.

Информация о состоянии устройств (занято или свободно) содержится в слове конфликта I.CONSIT (см.рис.9). Код конфликта для каждого устройства содержится в таблице конфликтов устройств I.CONFLC (см.табл.25).

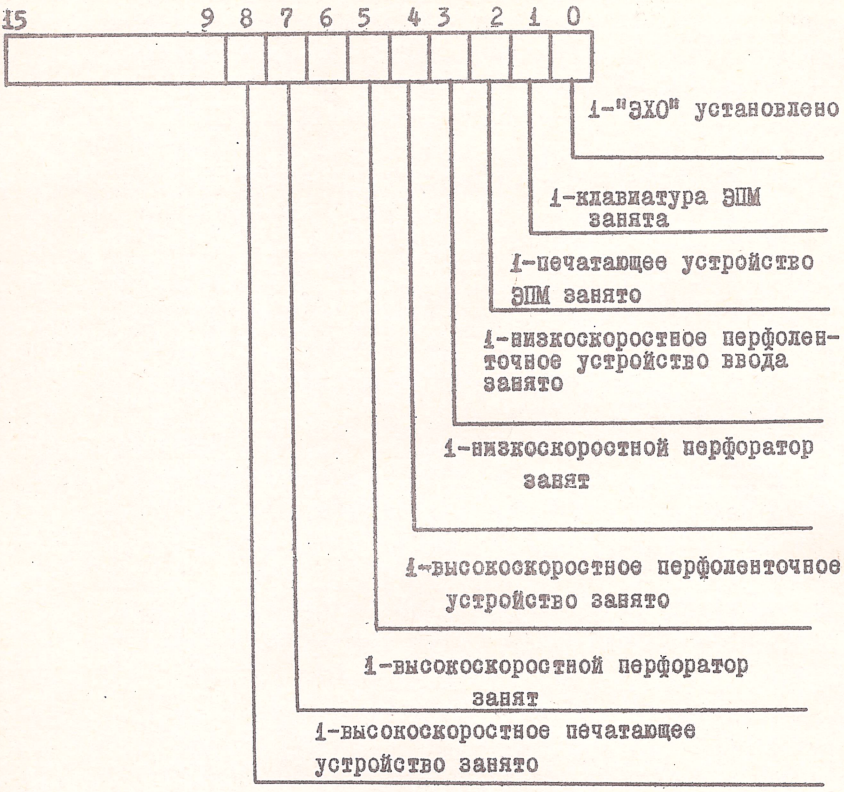


Рис.9. Структура слова конфликта

Таблица 25

Устройство	Конфликтующее устройство или операция	Код конфликта
Клавиатура ЭПМ	"ЭХО", клавиатура ЭПМ, печатающее устройство ЭПМ, низкоскоростное перфоленточное устройство ввода, низкоскоростной перфоратор	37
Печатающее устройство ЭПМ	"ЭХО", клавиатура ЭПМ, печатающее устройство ЭПМ, низкоскоростной перфоратор	27
Низкоскоростное перфоленточное устройство ввода	Клавиатура ЭПМ, низкоскоростное перфоленточное устройство ввода	12
Низкоскоростной перфоратор	"ЭХО", клавиатура ЭПМ, печатающее устройство ЭПМ, низкоскоростной перфоратор	27
Высокоскоростное перфоленточное устройство ввода	Высокоскоростное перфоленточное устройство ввода	40
Высокоскоростной перфоратор	Высокоскоростной перфоратор	100
Высокоскоростное печатающее устройство	Высокоскоростное печатающее устройство	400

Конфликт разрешается с помощью команды Ожидание.

8.4.4. Команда Ожидание

Формат команды

```
IOT
.WORD   <адрес возврата>
.BYTE  4,<номер позиции>
```

По команде Ожидание супервизор разрешает конфликт путем проверки состояния устройства. Если устройство свободно, управление переходит к команде, следующей за командой Ожидание. В противном случае супервизор передает управление ячейке, адрес которой указан в команде Ожидание. Если необходимо дождаться освобождения устройства, то во втором слове команды Ожидания записывается адрес самой команды Ожидание.

Помимо команды Ожидание для проверки окончания передачи данных используется проверка разряда Окончание.

Пример:

```
        Программа А
        READ = 11
        WRITE = 12
        IOT
        .WORD BUF2
        .BYTE READ, 2
        IOT
        .WORD BUF1
        .BYTE READ,2
        IOT
        .WORD BUF2
        .BYTE WRITE,3
DUNTST: TSTB  BUF1+3
        BPL DUNTST
```

```
        Программа В
        INIT = 1
        READ = 11
        WRITE = 12
        WAITR = 4
        IOT
        .WORD BUF2
        .BYTE  READ, 2
        IOT
        .WORD BUF1
        .BYTE  READ,2
        IOT
        .WORD BUF2
        .BYTE  WRITE,3
DEVTST: IOT
        .WORD DEVTST
        .BYTE WAITR,2
        IOT
        .WORD DEV
        .BYTE INIT,4
```

По программам А и В выполняется ввод с одного и того же устройства в два различных буфера.

8.4.5. Передача данных с использованием двух буферов

Передача данных может быть организована с использованием двух буферов. Это позволяет сократить время работы программы пользователя. Следующий пример показывает передачу данных с использованием двух буферов.

```
      Пример:
      READ=11
      IOT
      .WORD BUF1
      .BYTE READ,3 ; ВВОД В BUF1
A:    IOT
      .WORD BUF2
      .BYTE READ,3 ; ВВОД В BUF2
         :         ; ОБРАБОТКА ДАННЫХ BUF1
         :
B:    IOT
      .WORD BUF1
      .BYTE READ,3 ; ВВОД В BUF1
         :         ; ОБРАБОТКА ДАННЫХ BUF2
        JMP A
```

Данные в буфере 1 (BUF1) обрабатываются в то время, когда новые данные вводятся в буфер 2 (BUF2). И, соответственно, данные в буфере 2 обрабатываются в то время, когда новые данные вводятся в буфер 1. Супервизор проверяет занято' устройство или находится оно в конфликте и только после этого начинает ввод данных.

8.4.6. Команда Чтение РВ

Формат команды

```
IOT
.WORD      <адрес буфера>
.BYTE  13,    <номер позиции>
.WORD      <адрес выполнения>
```

Команда Чтение РВ аналогична команде Чтение, только по окончанию передачи данных супервизор передает управление программе пользователя по определенному адресу выполнения на приоритетном уровне устройства.

8.4.7. Команда Запись РВ

Формат команды:

```
IOT
.WORD       <адрес буфера>
.BYTE   14, <номер позиции>
.WORD   <адрес выполнения>
```

Команда Запись РВ аналогична команде Запись, только по окончанию передачи данных супервизор передает управление программе пользователя по определенному адресу выполнения на приоритетном уровне устройства.

8.5. Повторный пуск программы и возможность повторного считывания

8.5.I. Команда Повторный пуск

Формат команды:

```
IOT
.WORD   <адрес повторного пуска>
.BYTE 3,0
```

По команде Повторный пуск запоминается адрес повторного пуска программы пользователя. Для выполнения повторного пуска программы необходимо на клавиатуре ЭПМ нажать комбинацию клавиш СКЛ и Р (буква "Р" на нижнем регистре клавиатуры ЭПМ). Перед повторным пуском программы низкоскоростное перфоленточное устройство ввода должно быть выключено. Если адрес повторного пуска равен нулю, то СКЛ/Р игнорируется.

Примечание. После повторного пуска в программе пользователя должна быть предусмотрена установка указателя стека и всех периферийных устройств ввода-вывода в исходное состояние.

8.5.2. Команда Поиск

Формат команды

```
IOT
.WORD  0
.BYTE 5, <номер позиции для перфоленточного устройства
          ввода>
```

По команде Поиск очищается слово в таблице состояния устройств I.DST (см.п.8.7.2), делая возможным ввод с требуемого устройства.

8.6. Остановы

Ошибки, коды которых приведены в табл.26, вызывают останов программы по адресу 000040.

В случае останова R0 содержит код ошибки. Если код ошибки равен нулю, то в R1 находится содержимое СК, в противном случае - слово команды супервизора. В стек записывается содержимое регистров в следующем порядке: R5, R4, R3, R2, R1, R0, R7, R6

Таблица 26

Код ошибки	Причина
0	Обращение к адресу несуществующей памяти или переполнение стека
1	Недопустимая команда супервизора
2	Номер позиции > 7
3	Код устройства вне предела (1 ÷ 10)
4	Номер позиции не равен нулю
5	Неправильный режим передачи данных

8.7. Внутренние таблицы

8.7.I. Таблица прерывания устройства

В супервизоре для каждого устройства имеется своя таблица прерывания устройства, которая содержит необходимую информацию для обслуживания этого устройства (см.рис.IO).

Контрольная сумма
Длина буфера (целое без знака)
Адрес байта режима
Текущий адрес буфера
Счетчик байтов (целое без знака)
Код устройства
Адрес выполнения
Адрес регистра ввода

Рис.IO. Структура таблицы прерывания устройства

По завершении передачи данных устанавливается информация в заголовке буфера по данным таблицы прерывания устройства.

8.7.2. Таблица состояния устройств

Таблица состояния устройств I.DST содержит для каждого устройства слово, в котором устанавливается "1", если устройство выключено или нет перфоленты при вводе.

При передаче данных супервизор проверяет установку 1 - цы

в соответствующем слове. Если установлена, то в байте состояния супервизор устанавливает разряд Конец файла. Это возможно только для перфоленточных устройств ввода.

8.7.3. Добавление периферийных устройств

Для супервизора можно увеличить количество обслуживаемых периферийных устройств ввода-вывода. Для этого необходимо добавляемому устройству присвоить код устройства, расширить внутренние таблицы супервизора и включить в состав супервизора программу, обслуживающую добавляемое устройство.

8.8. Загрузка

Программа "Супервизор ввода-вывода" находится на перфоленте в виде объектного модуля.

Загрузочный модуль программы "Супервизор ввода-вывода" можно получить совместно с программой пользователя или отдельно.

В первом случае объектный модуль программы "Супервизор ввода-вывода" и объектные модули программы пользователя компонуются с помощью программы "Компоновщик" в единый загрузочный модуль. Полученный загрузочный модуль загружается в память с помощью Абсолютного загрузчика (см.п.6.2.).

Во втором случае загрузочный модуль программы "Супервизор ввода-вывода" получается отдельно и загружается в память с помощью Абсолютного загрузчика.

Коды ЭПМ CONSUL 260

П	КОД	ЗНАК	П	КОД	ЗНАК	П	КОД	ЗНАК	П	КОД	ЗНАК
0	000	ПУС	1	040	Пробел	1	100		0	140	Ю
0	001		0	041	!	0	101	A	1	141	А
1	002		0	042	"	0	102	B	1	142	Б
0	003		1	043	#	1	103	C	0	143	Ц
1	004		0	044	§	0	104	D	1	144	Д
0	005		1	045	%	1	105	E	0	145	Е
0	006		1	046	&	1	106	F	0	146	Ф
1	007		0	047	'	0	107	G	1	147	Г
1	010	БШ	0	050	(	0	110	H	1	150	Х
0	011	ГТ	1	051	)	1	111	I	0	151	И
0	012	ПС	1	052	*	1	112	J	0	152	Й
1	013		0	053	+	0	113	K	1	153	К
0	014	ПФ	1	054	,	1	114	L	0	154	Л
1	015	ВК	0	055	-	0	115	M	1	155	М
1	016	ВР	0	056	.	0	116	N	1	156	Н
0	017	НР	1	057	/	1	117	O	0	157	О
1	020		0	060	0	0	120	P	1	160	П
0	021	КЛ	1	061	1	1	121	Q	0	161	Я
0	022	ЧЛ	1	062	2	1	122	R	0	162	Р
1	023		0	063	3	0	123	S	1	163	С
0	024		1	064	4	1	124	T	0	164	Т
1	025		0	065	5	0	125	U	1	165	У
1	026		0	066	6	0	126	V	1	166	Ж
0	027		1	067	7	1	127	W	0	167	В
0	030		1	070	8	1	130	X	0	170	Ь
1	031		0	071	9	0	131	Y	1	171	Ы
1	032		0	072	:	0	132	Z	1	172	З
0	033		1	073	;	1	133	[	0	173	Ш
1	034		0	074	<	0	134	\	1	174	Э
0	035		1	075	=	1	135	]	0	175	Щ
0	036		1	076	>	1	136	¬	0	176	Ч
1	037		0	077	?	0	137	_	1	177	ЗБ

В данной таблице используются следующие сокращения:

П - проверочный элемент
ПУС - пусто
ВШ - возврат на шаг
ГТ - горизонтальная табуляция
ПС - перевод строки
ПФ - перевод формата
ВК - возврат каретки
ВР - верхний регистр
НР - нижний регистр
КЛ - красная лента
ЧЛ - черная лента
ЗБ - забой

Коды управляющих символов (0-37) получаются одновременным нажатием клавиши СКЛ и соответствующей клавиши 3-го столбца (коды 100-137) данной таблицы.

Перемещающий ассемблер и Ассемблер

I. СПЕЦИАЛЬНЫЕ ЗНАКИ

Таблица I

Используемый знак	Назначение
ПФ	Ограничитель строки
ПС	Ограничитель строки
ВК	Ограничитель оператора
:	Ограничитель метки
=	Признак присваивания
%	Признак регистра
ГТ	Ограничитель поля
пробел	Ограничитель поля
#	Непосредственная адресация
@	Косвенная адресация
(	Начальный указатель регистра
)	Конечный указатель регистра
,	Разделитель операндов
;	Указатель поля комментария
+	Знак операции сложения
-	Знак операции вычитания
&	Знак логической операции "И"
!	Знак логической операции "ИЛИ"
"	Преобразователь двух символов
'	Преобразователь одного символа
.	Счетчик адресов

Таблица 2

Формат команды	Адресация	Метод адресации
R	0N	Регистровый
@R или (ER)	1N	Косвенно-регистровый
(ER)+	2N	Автоинкрементный
@(ER)+	3N	Косвенно-автоинкрементный
-(ER)	4N	Автодекрементный
@-(ER)	5N	Косвенно-автодекрементный
E(ER)	6N	Индексный
@E(ER)	7N	Косвенно-индексный
# E	27	Непосредственный
@ # E	37	Абсолютный
E	67	Относительный
@E	77	Косвенно-относительный

В табл. 2 N определяет используемый регистр.

3. СИСТЕМА КОМАНД

Таблица 3

Двухадресные команды

Мнемоника	Операция	Код команды	Признаки N Z V C
MOV(B)	Пересылка (байта)	.1SSDD	* * 0 -
CMP(B)	Сравнение (байтов)	.2SSDD	* * * *
BIT(B)	Проверка разрядов(байта)	.3SSDD	* * 0 -
BIC(B)	Очистка разрядов (байта)	.4SSDD	* * 0 -
BIS(B)	Логическое сложение(байтов)	.5SSDD	* * 0 -
ADD	Сложение	06SSDD	* * * *
SUB	Вычитание	16SSDD	* * * *

Таблица 4

Одноадресные команды

Мнемоника	Операция	Код команды	Признаки N Z V C
CLR(B)	Очистка (байта)	.050DD	0 1 0 0
COM(B)	Инвертирование (байта)	.051DD	* * 0 1
INC(B)	Прибавление единицы (к байту)	.052DD	* * * -
DEC(B)	Вычитание единицы (из байта)	.053DD	* * * -
NEG(B)	Изменение знака	.054DD	* * * *
ADC(B)	Прибавление переноса (к байту)	.055DD	* * * *
SBC(B)	Вычитание переноса (из байта)	.056DD	* * * *
TST(B)	Проверка (байта)	.057DD	* * 0 0
ROR(B)	Циклический сдвиг вправо (байта)	.060DD	* * * *

Продолжение табл.4

Мнемоника	Операция	Код команды	Признаки N Z V C
ROL(B)	Циклический сдвиг влево (байта)	.061DD	* * * *
ASR(B)	Арифметический сдвиг вправо (байта)	.062DD	* * * *
ASL(B)	Арифметический сдвиг влево (байта)	.063DD	* * * *
JMP	Безусловный переход	0001DD	- - - -
SWAB	Перестановка байтов	0003DD	* * 0 0

Таблица 5

Команды управления

Мнемоника	Операция	Код команды	Признаки N Z V C
HALT	Останов	000000	- - - -
WAIT	Ожидание	000001	- - - -
RTI	Возврат из прерывания	000002	* * * *
	Командное прерывание для отладки	000003	* * * *
IOT	Командное прерывание для ввода-вывода	000004	* * * *
RESET	Сброс периферийных устройств	000005	- - - -
EMT	Командное прерывание для системных программ	104000÷ 104377	* * * *
TRAP	Командное прерывание	104400÷ 104777	* * * *

Команды изменения признаков

Таблица 6

Мнемоника	Операция	Код команды
CLC	Очистка C	000241
SEC	Установка C	000261
CLV	Очистка V	000242
SEV	Установка V	000262
CLZ	Очистка Z	000244
SEZ	Установка Z	000264
CLN	Очистка N	000250
SEN	Установка N	000270
CNZ	Очистка N и Z	000254
CCC	Очистка всех признаков	000257
SCC	Установка всех признаков	000277
NOP	Нет операции	000240

Команды ветвления

Таблица 7

Мнемоника	Операция	Код команды	Признаки N Z V C
BR	Ветвление безусловное	0004XX	- - - -
BNE	Ветвление, если не равно (нулю)	0010XX	- - - -
BEQ	Ветвление, если равно (нулю)	0014XX	- - - -
BGE	Ветвление, если больше или равно (нулю)	0020XX	- - - -
BLT	Ветвление, если меньше (нуля)	0024XX	- - - -
BGT	Ветвление, если больше (нуля)	0030XX	- - - -
BLE	Ветвление, если меньше или равно (нулю)	0034XX	- - - -
BPL	Ветвление, если плюс	1000XX	- - - -
BMI	Ветвление, если минус	1004XX	- - - -
BHI	Ветвление, если больше	1010XX	- - - -
BLOS	Ветвление, если меньше или равно	1014XX	- - - -
BVC	Ветвление, если нет арифметического переполнения	1020XX	- - - -

Мнемоника	Операция	Код команды	Признаки N Z V C
BVS	Ветвление, если арифметическое переполнение	1024XX	- - - -
BCC	Ветвление, если нет переноса	1030XX	- - - -
BCS	Ветвление, если перенос	1034XX	- - - -
BHIS	Ветвление, если больше или равно	1030XX	- - - -
BLO	Ветвление, если меньше	1034XX	- - - -

Таблица 8

Команды обращения к подпрограмме и возврата из подпрограммы

Мнемоника	Операция	Код команды	Признаки N Z V C
JSR	Обращение к подпрограмме	004RDD	- - - -
RTS	Возврат из подпрограммы	00020R	- - - -

Таблица 9

Команды для Перемещающего ассемблера

Мнемоника	Операция	Код команды	Признаки N Z V C
SXT	Расширение знака	0067DD	- * - -
MFPS	Чтение ССП	1067DD	* * 0 -
MTPS	Запись ССП	1064DD	* * * *
XOR	Исключающее ИЛИ	074RDD	* * 0 -
MUL	Умножение	070RDD	* * 0 *
DIV	Деление	071RSS	* * * *
ASH	Арифметический сдвиг	072RSS	* * * *
ASHC	Арифметический сдвиг двойного слова	073RSS	* * * *
MARK	Восстановление УС	0064NN	- - - -

Продолжение табл.9

Мнемоника	Операция	Код команды	Признаки N Z V C
SOB	Вычитание единицы и ветвление	077RNN	- - - -
RTT	Возврат из прерывания	000006	* * * *

Команды MFPS и MTPS используются для ЭВМ, не имеющей КР.

В табл.3-9 используются следующие обозначения:

В графе "Операция"

N - признак отрицательности результата
Z - признак равенства нулю результата
V - признак арифметического переполнения
C - признак переноса из старшего разряда

В графе "Код команды"

. - разряд слово/байт устанавливается в "1" для байтовых команд (100000)
SS - поле адресации операнда источника
DD - поле адресации операнда приемника
XX - смещение (8 разрядов)
NN - смещение (6 разрядов)
R - регистр общего назначения (0-7)

В графе "Признаки"

* - устанавливается в зависимости от выполнения условий
- - не изменяется
0 - очищается
1 - устанавливается

Таблица 10

Директивы Перемещающего ассемблера и Ассемблера

Мнемоника	Действие	Пояснение
.EOT	Конец ленты	Указывает на физический конец исходной ленты
.EVEN	Четность	Обеспечивает четность счетчика адресов

Продолжение табл.IO

Мнемоника	Действие	Пояснение
.ENDm	Конец	Указывает на физический и логический конец исходного модуля (m - пусковой адрес программы)
.WORD	Слово	Используется для записи данных в виде последовательно расположенных слов
.BYTE	Байт	Используется для записи данных в виде последовательно расположенных байтов
.ASCII	Преобразование символов	Используется для преобразования символов в их 7-битные коды

Таблица II

Директивы Перемещающего ассемблера

Мнемоника	Действие	Пояснение
.TITLE	Наименование	Используется для присвоения наименования объектному модулю
.GLOBL	Глобальное имя	Используется для описания имен как глобальных
.ASECT .CSECT	Секция	Указывает на абсолютную или относительную секцию программы
.RAD 50	Преобразования символов	Используется для преобразования символов в форму RAD50
.LIMIT	Граница	Резервирует две ячейки памяти для запоминания младшего и старшего адреса относительного кода загрузочного модуля
.IFZ .IFNZ .IFL .IFLE .IFG .IFGE .IFDF .IFNDF	Директивы условной трансляции	Позволяют включать или исключать из процесса трансляции участки исходного текста в соответствии с условием директивы

4. КОДЫ RAD50

Таблица I2

Символ	Код символа	Код RAD50
Пробел	40	0
A - Z	101 - 132	1-32
$	44	33
.	56	34
0 - 9	60 - 71	36-47

Таблица 13

Символ	Код одиночного или первого символа	Код второго символа	Код третьего символа
A	003100	000050	000001
B	006200	000120	000002
C	011300	000170	000003
D	014400	000240	000004
E	017500	000310	000005
F	022600	000360	000006
G	025700	000430	000007
H	031000	000500	000010
I	034100	000550	000011
J	037200	000620	000012
K	042300	000670	000013
L	045400	000740	000014
M	050500	001010	000015
N	053600	001060	000016
O	056700	001130	000017
P	062000	001200	000020
Q	065100	001250	000021
R	070200	001320	000022
S	073300	001370	000023
T	076400	001440	000024
U	101500	001510	000025

Продолжение табл.13

Символ	Код одиночного или первого символа	Код второго символа	Код третьего символа
V	104600	001560	000026
W	107700	001630	000027
X	113000	001700	000030
Y	116100	001750	000031
Z	121200	002020	000032
$	124300	002070	000033
.	127400	002140	000034
0	135600	002260	000036
1	140700	002330	000037
2	144000	002400	000040
3	147100	002450	000041
4	152200	002520	000042
5	155300	002570	000043
6	160400	002640	000044
7	163500	002710	000045
8	166600	002760	000046
9	171700	003030	000047

Таблица 13 позволяет преобразовать символы в коды RAD50.

Пример:

.RAD/X2B/

в коде RAD50 получим следующее значение (арифметические действия выполняются в восьмеричной системе):

```
    X = 113000
+   2 = 002400
    B = 000002
-------------
  X2B = 115402
```

Приложение 3

Команды Редактора текста

Таблица

Команда	Формат команды	Назначение
Открыть	O	Команда аналогична команде "Поиск" программы "Супервизор ввода-вывода"
Чтение	R	Считать страницу текста с перфоленты в буфер страниц
Печать	OL nL @L /L	Печать строк, количество которых определяется аргументом
	V	Печать текущей строки
	<	Печать предшествующей строки
	>	Печать следующей строки
Перфорация	OP nP @P /P	Перфорация строк, количество которых определяется аргументом
Следующий	nN	Перфорация содержимого буфера страниц (и, если последним знаком буфера является знак ПФ, перфорация зоны синхродорожки), стирание содержимого буфера страниц, считывание следующей страницы. Эта последовательность действий выполняется n раз.
Перевод формата	F	Перфорация кода знака ПФ и зоны синхродорожки
Конец ленты	nT	Перфорация n зон синхродорожки
Начало	B	Перемещение "точки" в начало буфера страниц
Конец	E	Перемещение "точки" в конец буфера страниц
Переход	OJ nJ @J /J	Перемещение "точки" на число знаков, определяемое аргументом
Продвижение	OA nA	Перемещение "точки" на число строк, определяемое аргументом

Продолжение таблицы

Команда	Формат команды	Назначение
	@A /A	
Маркер	M	Запоминание положения "точки"
Поиск	nG	Поиск текста, встречающегося в n-ый раз в буфере страниц
Поиск общий	H	Поиск текста во всем массиве текста
Вставка	I	Вставка текста
Стирание знаков	0D nD @D /D	Стирание числа знаков, определяемого аргументом
Стирание строк	0K nK @K /K	Стирание числа строк, определяемого аргументом
Замена знаков	0C nC @C /C	Замена числа знаков, определяемого аргументом
Замена строк	0X nX @X /X	Замена числа строк, определяемого аргументом

Приложение 4

Команды Программы отладки

Таблица

Формат команды	Наименование команды
n/	Открыть слово в ячейке *n*
/	Открыть слово, которое было открыто последним
n\	Открыть байт в ячейке *n*
\	Открыть байт, который был открыт последним
↓	Открыть следующую ячейку
↑	Открыть предыдущую ячейку
↓	Закрыть ячейку
@	Интерпретировать содержимое открытой ячейки как абсолютный адрес и открыть ячейку по этому адресу
_	Интерпретировать содержимое открытой ячейки как индексное слово относительного метода адресации и открыть ячейку по исполнительному адресу
>	Интерпретировать младший байт открытой ячейки как смещение команды ветвления и открыть ячейку определяемую этим смещением
<	Открыть ячейку, следующую за ячейкой, к которой была подана команда ">", "@" или "_"
%*n*/	Открыть регистр общего назначения *n* (*n* =0-7)
%S/	Открыть регистр состояния программы
%P/	Открыть ячейку, содержащую приоритет ПО
n;B	Задать точку останова *n*
n;*m*B	Задать точку останова *n* с номером *m*
;B	Удалить все точки останова
;*m*B	Удалить точку останова с номером *m*
%B/	Определить точку останова с номером 0
n;G	Запустить программу с адреса *n*
;P	Продолжить Выполнение программы с точки останова
K;P	Продолжить выполнение программы с точки останова к раз
%M/	Открыть маску поиска
n;W	Поиск слова *n*
n;E	Поиск адреса *n*
n/x*m*;O	Вычислить смещение от ячейки *n* до ячейки *m*
;*n*S	Установить командный режим
;S	Выйти из командного режима

Дублирование перфолент

Программа "Дублирование перфолент" предназначена для получения копий перфолент на высокоскоростных или низкоскоростных перфоленточных устройствах ввода-вывода.

Ввод программы в память ЭВМ осуществляется вручную с помощью переключателей КР на ЭВМ с клавишным регистром или с клавиатуры ЭПМ на ЭВМ без клавишного регистра.

Ниже приводится текст программы "Дублирование перфолент" с адреса 000000.

```
000000  016700
000002  000024
000004  016701
000006  000022
000010  005210
000012  105710
000014  100376
000016  105711
000020  100376
000022  022021
000024  111011
000026  000764
000030  177550 }  высокоскоростные   или  177560 }  низкоскорост-
000032  177554 }  устройства             177564 }  ные устройства
```

В ячейки 000030, 000032 вводятся адреса регистров состояний используемых перфоленточных устройств ввода-вывода.

Дублирование исходной перфоленты осуществляется следующим образом:

а) заправить исходную перфоленту в используемое перфоленточное устройство ввода и включить его питание;

б) включить питание используемого перфоратора;

в) запустить программу с адреса загрузки. Исходная лента считывается в устройстве ввода, на перфоратор выдается дубликат.

Лист регистрации изменений

Изм.	Номера листов				Всего листов в докумен-те	№ докум.	Входящий № сопро-водитель-ного до-кумента и дата	Под-пись	Да-та
	Изме-ненных	Заме-ненных	Но-вых	Аннули-рован-ных					

СОДЕРЖАНИЕ

ВВЕДЕНИЕ 3
1. ПЕРЕМЕЩАЮЩИЙ АССЕМБЛЕР 6
2. АССЕМБЛЕР 41
3. КОМПОНОВЩИК 68
4. РЕДАКТОР ТЕКСТА 78
5. ПРОГРАММА ОТЛАДКИ 91
6. ПРОГРАММЫ ЗАГРУЗКИ 106
7. ПРОГРАММЫ ВЫВОДА 118
8. СУПЕРВИЗОР ВВОДА-ВЫВОДА 124
Приложение 1 141
Приложение 2 143
Приложение 3 153
Приложение 4 155
Приложение 5 156

Формат 60×90 1/16 Тираж 1000 экз. Зак. 368

Производственно-издательский отдел ЦНИИ "Электроника"